AAMAS 2012

The 11th International Conference on Autonomous Agents and Multiagent Systems

June 4—8, 2012

Valencia, Spain

Proceedings

Volume II

IFAAMAS

IFAAMAS
International Foundation for Autonomous Agents and Multiagent Systems
`www.ifaamas.org`

ISBN-10: 0-9817381-2-5
ISBN-13: 978-0-9817381-2-3

Contents

Invited Talks

Main Program - Full Papers

Session 1A – Innovative Applications

PROTECT: A Deployed Game Theoretic System to Protect the Ports of the United States
Eric Shieh, Bo An, Rong Yang, Milind Tambe, Craig Baldwin, Joseph DiRenzo, Ben Maule, Garrett Meyer 13

SAVES: A Sustainable Multiagent Application to Conserve Building Energy Considering Occupants
Jun-young Kwak, Pradeep Varakantham, Rajiv Maheswaran, Milind Tambe, Farrokh Jazizadeh, Geoffrey Kavulya, Laura Klein, Burcin Becerik-Gerber, Timothy Hayes, Wendy Wood 21

Active Malware Analysis using Stochastic Games
Simon Williamson, Pradeep Varakantham, Debin Gao, Ong Chen Hui 29

Agents vs. Pirates: Multi-agent Simulation and Optimization to Fight Maritime Piracy
Michal Jakob, Ondřej Vaněk, Ondřej Hrstka, Michal Pěchouček 37

Improving Building Energy Efficiency with a Network of Sensing, Learning and Prediction Agents
Sunil Mamidi, Yu-Han Chang, Rajiv Maheswaran 45

Session 2A – Virtual Agents

Bayesian Model of the Social Effects of Emotion in Decision-Making in Multiagent Systems
Celso de Melo, Peter Carnevale, Stephen Read, Dimitrios Antos, Jonathan Gratch 55

Towards building a Virtual Counselor: Modeling Nonverbal Behavior during Intimate Self-Disclosure
Sin-Hwa Kang, Jonathan Gratch, Candy Sidner, Ron Artstein, Lixing Huang, Louis-Phillippe Morency 63

A Sequential Recommendation Approach for Interactive Personalized Story Generation
Hong Yu, Mark Riedl 71

Evaluating the Models & Behaviour of 3D Intelligent Virtual Animals in a Predator-Prey Relationship
Deborah Richards, Michael J. Jacobson, John Porte, Charlotte Taylor, Meredith Taylor, Anne Newstead, Iwan Kelaiah, Nader Hanna 79

Model of the Perception of Smiling Virtual Character
Magalie Ochs, Catherine Pelachaud 87

Session 3A – Robotics I

Supervised Morphogenesis - Morphology Control of Ground-based Self-Assembling Robots by Aerial Robots
Nithin Mathews, Alessandro Stranieri, Alexander Scheidler, Marco Dorigo 97

Decentralized Active Robotic Exploration and Mapping for Probabilistic Field Classification in Environmental Sensing
Kian Hsiang Low, Jie Chen, John Dolan, Steve Chien, David Thompson 105

Robot Exploration with Fast Frontier Detection: Theory and Experiments
Matan Keidar, Gal Kaminka 113

Dynamic Reconfiguration in Modular Robots using Graph Partitioning-based Coalitions
Prithviraj Dasgupta, Vladimir Ufimtsev, Carl Nelson, S. G. M. Hossain 121

UT Austin Villa 2011: A Champion Agent in the RoboCup 3D Soccer Simulation Competition
Patrick MacAlpine, Daniel Urieli, Samuel Barrett, Shivaram Kalyanakrishnan, Francisco Barrera, Adrian Lopez-Mobilia, Nicolae Ştiurcă, Victor Vu, Peter Stone 129

Session 4A – Robotics II

Property-driven design for swarm robotics
Manuele Brambilla, Carlo Pinciroli, Mauro Birattari, Marco Dorigo 139

Multi-robot collision avoidance with localization uncertainty
Daniel Hennes, Daniel Claes, Wim Meeussen, Karl Tuyls 147

Decision-Theoretic Approach to Maximizing Observation of Multiple Targets in Multi-Camera Surveillance
Prabhu Natarajan, Trong Nghia Hoang, Kian Hsiang Low, Mohan Kankanhalli 155

Segregation in Swarms of e-puck Robots Based On the Brazil Nut Effect
Jianing Chen, Melvin Gauci, Michael J. Price, Roderich Groß 163

Model-Driven Behavior Specification for Robotic Teams
Alexandros Paraschos, Nikolaos Spanoudakis, Michail Lagoudakis 171

Session 5A – Robotics III

Active Visual Sensing and Collaboration on Mobile Robots using Hierarchical POMDPs
Shiqi Zhang, Mohan Sridharan 181

What am I doing? Automatic Construction of an Agent's State-Transition Diagram through Introspection
Constantin Berzan, Matthias Scheutz 189

Learning from Demonstration with Swarm Hierarchies
Keith Sullivan, Sean Luke 197

Autonomous Robot Dancing Driven by Beats and Emotions of Music
Guangyu Xia, Junyun Tay, Roger Dannenberg, Manuela Veloso 205

Session 1B – Teamwork I

Coordination Guided Reinforcement Learning
Qiangfeng Peter Lau, Mong Li Lee, Wynne Hsu 215

On Coalition Formation with Sparse Synergies
Thomas Voice, Sarvapali Ramchurn, Nick Jennings 223

Decentralised Channel Allocation and Information Sharing for Teams of Cooperative Agents
Sebastian Stein, Simon Williamson, Nick Jennings 231

A New Approach to Betweenness Centrality Based on the Shapley Value
Piotr Szczepański, Tomasz Michalak, Talal Rahwan 239

Maintaining Team Coherence under the Velocity Obstacle Framework
Andrew Kimmel, Andrew Dobson, Kostas Bekris 247

Session 2B – Distributed Problem Solving

Stochastic Dominance in Stochastic DCOPs for Risk Sensitive Applications
Duc Thien Nguyen, William Yeoh, Hoong Chuin Lau 257

Max/Min-sum Distributed Constraint Optimization through Value Propagation on an Alternating DAG
Roie Zivan, Hilla Peled 265

Improving BnB-ADOPT$^+$-AC
Patricia Gutierrez, Pedro Meseguer 273

Optimal Decentralised Dispatch of Embedded Generation in the Smart Grid
Sam Miller, Sarvapali Ramchurn, Alex Rogers 281

DCOPs and Bandits: Exploration and Exploitation in Decentralised Coordination
Ruben Stranders, Long Tran-Thanh, Francesco Maria Delle Fave, Alex Rogers, Nick Jennings . . 289

Session 4B – Agent Societies

A Multiagent Evolutionary Framework based on Trust for Multiobjective Optimization
Siwei Jiang, Jie Zhang, Yew-Soon Ong 299

A qualitative reputation system for multiagent systems with protocol-based communication
Emilio Serrano, Michael Rovatsos, Juan Botia 307

PRep: A Probabilistic Reputation Model for Biased Societies
Yasaman Haghpanah, Marie desJardins 315

A Decision-Theoretic Characterization of Organizational Influences
Jason Sleight, Ed Durfee 323

Reasoning under Compliance Assumptions in Normative Multiagent Systems
Max Knobbout, Mehdi Dastani 331

Session 5B – Teamwork II

Leading Ad Hoc Agents in Joint Action Settings with Multiple Teammates
Noa Agmon, Peter Stone 341

Comparative Evaluation of MAL Algorithms in a Diverse Set of Ad Hoc Team Problems
Stefano Albrecht, Subramanian Ramamoorthy 349

An Analysis Framework for Ad Hoc Teamwork Tasks
Samuel Barrett, Peter Stone 357

Modeling and Learning Synergy for Team Formation with Heterogeneous Agents
Somchaya Liemhetcharat, Manuela Veloso 365

Session 1C – Learning I

V-MAX: Tempered Optimism for Better PAC Reinforcement Learning
Karun Rao, Shimon Whiteson 375

Reinforcement Learning Transfer via Sparse Coding
Haitham Bou Ammar, Karl Tuyls, Matthew Taylor, Kurt Driessen, Gerhard Weiss 383

Learning in a Small World
Arun Tejasvi Chaganty, Prateek Gaur, Balaraman Ravindran 391

Just Add Pepper: Extending Learning Algorithms for Repeated Matrix Games to Repeated Markov Games
Jacob Crandall 399

Strong Mitigation: Nesting Search for Good Policies Within Search for Good Reward
Jeshua Bratman, Satinder Singh, Richard Lewis, Jonathan Sorg 407

Session 2C – Learning II

Decentralized Bayesian Reinforcement Learning for Online Agent Collaboration
Luke Teacy, Georgios Chalkiadakis, Alessandro Farinelli, Alex Rogers, Nick Jennings, Sally McClean, Gerard Parr 417

Shaping Fitness Functions for Coevolving Cooperative Multiagent Systems
Mitchell Colby, Kagan Tumer 425

Dynamic Potential-Based Reward Shaping
Sam Devlin, Daniel Kudenko 433

Learning and Predicting Dynamic Networked Behavior with Graphical Multiagent Models
Quang Duong, Michael Wellman, Satinder Singh, Michael Kearns 441

Session 3C – Human-agent Interaction

A Cultural Sensitive Agent for Human-Computer Negotiation
Galit Haim, Ya'akov (Kobi) Gal, Sarit Kraus, Michele Gelfand 451

Giving Advice to People in Path Selection Problems
Amos Azaria, Zinovi Rabinovich, Sarit Kraus, Claudia Goldman, Omer Tsimhoni 459

Combining Human and Machine Intelligence in Large-scale Crowdsourcing
Ece Kamar, Severin Hacker, Eric Horvitz 467

Reinforcement Learning from Simultaneous Human and MDP Reward
W. Bradley Knox, Peter Stone 475

Automatic Task Decomposition and State Abstraction from Demonstration
Luis C. Cobo, Charles L. Isbell Jr., Andrea Thomaz 483

Session 4C – Argumentation & Negotiation

Quantifying Disagreement in Argument-based Reasoning
Richard Booth, Martin Caminada, Mikolaj MikoŁaj, Iyad Rahwan 493

Cooperative Dialogues with Conditional Arguments
Samy Sá, João Alcântara 501

Defeasible Argumentation for Multi-Agent Planning in Ambient Intelligence Applications
Sergio Pajares Ferrando, Eva Onaindia 509

Personalizing Communication about Trust
Andrew Koster, Jordi Sabater-Mir, Marco Schorlemmer 517

From axiomatic to strategic models of bargaining with logical beliefs and goals
Bao Vo, Minyi Li 525

Session 5C – Emergence

Crowd IQ - Aggregating Opinions to Boost Performance
Yoram Bachrach, Thore Graepel, Gjergji Kasneci, Michal Kosinski, Jurgen Van-Gael 535

Efficient Opinion Sharing in Large Decentralised Teams
Oleksandr Pryymak, Alex Rogers, Nick Jennings 543

Agents of Influence in Social Networks
Amer Ghanem, Srinivasa Vedanarayanan, Ali Minai 551

The Emergence of Commitments and Cooperation
The Anh Han , Luís Moniz Pereira , Francisco C. Santos 559

Session 1D – Social Choice I

Strategyproof Approximations of Distance Rationalizable Voting Rules
Travis Service, Julie Adams 569

Campaigns for Lazy Voters: Truncated Ballots
Dorothea Baumeister, Piotr Faliszewski, Jérôme Lang, Jörg Rothe 577

Possible and Necessary Winners of Partial Tournaments
Haris Aziz, Markus Brill, Felix Fischer, Paul Harrenstein, Jérôme Lang, Hans Georg Seedig . . . 585

Communication Complexity of Approximating Voting Rules
Travis Service, Julie Adams 593

Session 2D – Social Choice II

Lot-based Voting Rules
Toby Walsh, Lirong Xia 603

Convergence of Iterative Voting
Omer Lev, Jeffrey Rosenschein 611

Optimal Manipulation of Voting Rules
Svetlana Obraztsova, Edith Elkind 619

Manipulation Under Voting Rule Uncertainty
Edith Elkind, Gábor Erdélyi 627

Voter Response to Iterated Poll Information
Annemieke Reijngoud, Ulle Endriss 635

Session 3D – Economies & Markets I

Rational Market Making with Probabilistic Knowledge
Abraham Othman, Tuomas Sandholm 645

Can a Zero-Intelligence Plus Model Explain the Stylized Facts of Financial Time Series Data?
Imon Palit, Steve Phelps, Wing Lon Ng 653

A Scoring Rule-based Mechanism for Aggregate Demand Prediction in the Smart Grid
Harry Rose, Alex Rogers, Enrico Gerding 661

A Model-Based Online Mechanism with Pre-Commitment and its Application to Electric Vehicle Charging
Sebastian Stein, Enrico Gerding, Valentin Robu, Nick Jennings 669

Efficient Crowdsourcing Contests
Ruggiero Cavallo, Shaili Jain 677

Session 4D – Economies & Markets II

Identifying Influential Agents for Advertising in Multi-agent Markets
Mahsa Maghami, Gita Sukthankar 687

Predicting Your Own Effort
David F. Bacon, Yiling Chen, Ian Kash, David Parkes, Malvika Rao, Manu Sridharan 695

Optimal Incentive Timing Strategies for Product Marketing on Social Networks
Pankaj Dayama, Aditya Karnik, Yadati Narahari 703

Optimizing Kidney Exchange with Transplant Chains: Theory and Reality
John Dickerson, Ariel Procaccia, Tuomas Sandholm 711

Fair Allocation Without Trade
Avital Gutman, Noam Nisan 719

Session 5D – Auction & Mechanism Design

Mixed-bundling auctions with reserve prices
Pingzhong Tang, Tuomas Sandholm 729

Eliciting Forecasts from Self-interested Experts: Scoring Rules for Decision Makers
Craig Boutilier 737

Worst-Case Optimal Redistribution of VCG Payments in Heterogeneous-Item Auctions with Unit Demand
Mingyu Guo 745

False-name-proofness in Online Mechanisms
Taiki Todo, Takayuki Mouri, Atsushi Iwasaki, Makoto Yokoo 753

Session 1E – Game Theory I

Existence of Stability in Hedonic Coalition Formation Games
Haris Aziz, Florian Brandl 763

Stablity Scores: Measuring Coalitional Stability
Michal Feldman, Reshef Meir, Moshe Tennenholtz 771

Coalitional Stability in Structured Environments
Georgios Chalkiadakis, Vangelis Markakis, Nick Jennings 779

Overlapping Coalition Formation Games: Charting the Tractability Frontier
Yair Zick, Georgios Chalkiadakis, Edith Elkind 787

Handling Negative Value Rules in MC-net-based Coalition Structure Generation
Suguru Ueda, Takato Hasegawa, Naoyuki Hashimoto, Naoki Ohta, Atsushi Iwasaki, Makoto Yokoo 795

Session 2E – Game Theory II

Short Sight in Extensive Games
Davide Grossi, Paolo Turrini 805

New Results on the Verification of Nash Refinements for Extensive-Form Games
Nicola Gatti, Fabio Panozzo 813

Playing Repeated Stackelberg Games with Unknown Opponents
Janusz Marecki, Gerry Tesauro, Richard Segal 821

Repeated zero-sum games with budget
Troels Sørensen 829

Efficient Nash Equilibrium Approximation through Monte Carlo Counterfactual Regret Minimization
Michael Johanson, Nolan Bard, Marc Lanctot, Richard Gibson, Michael Bowling 837

Session 3E – Game Theory III

Computing Optimal Strategy against Quantal Response in Security Games
Rong Yang, Fernando Ordóñez, Milind Tambe 847

A Unified Method for Handling Discrete and Continuous Uncertainty in Bayesian Stackelberg Games
Zhengyu Yin, Milind Tambe 855

Multi-Objective Optimization for Security Games
Matthew Brown, Bo An, Christopher Kiekintveld, Fernando Ordóñez, Milind Tambe 863

Strategy Purification and Thresholding: Effective Non-Equilibrium Approaches for Playing Large Games
Sam Ganzfried, Tuomas Sandholm, Kevin Waugh 871

Solving Non-Zero Sum Multiagent Network Flow Security Games with Attack Costs
Steven Okamoto, Noam Hazon, Katia Sycara 879

Session 4E – Game Theory IV

Task Routing for Prediction Tasks
Haoqi Zhang, Eric Horvitz, Yiling Chen, David Parkes 889

Mastering multi-player games
Yossi Azar, Uriel Feige, Michal Feldman, Moshe Tennenholtz 897

Game-theoretic Resource Allocation for Malicious Packet Detection in Computer Networks
Ondřej Vaněk, Zhengyu Yin, Manish Jain, Branislav Bošanský, Milind Tambe, Michal Pěchouček 905

Sustaining Cooperation on Networks: An Analytical Study based on Evolutionary Game Theory
Raghunandan Ananthasayanam, Subramanian Chandrasekarapuram 913

Behavioral Game Theoretic Models: A Bayesian Framework For Parameter Analysis
James Wright, Kevin Leyton-Brown 921

Session 5E – Game & Agent Theories

Scaling Simulation-Based Game Analysis through Deviation-Preserving Reduction
Bryce Wiedenbeck, Michael Wellman 931

Towards Tractable Boolean Games
Paul Dunne, Michael Wooldridge 939

A Framework for Modeling Population Strategies by Depth of Reasoning
Michael Wunder, Michael Kaisers, John Robert Yaros, Michael Littman 947

Detection of Suspicious Behavior from a Sparse Set of Multiagent Interactions
Boštjan Kaluža, Gal Kaminka, Milind Tambe 955

Session 1F – Planning

Probabilistic Planning with Non-Linear Utility Functions and Worst-Case Guarantees
Stefano Ermon, Carla Gomes, Bart Selman, Alexander Vladimirsky 965

Heuristic Search of Multiagent Influence Space
Stefan Witwicki, Frans Oliehoek, Leslie Kaelbling 973

A Hierarchical Goal-Based Formalism and Algorithm for Single-Agent Planning
Vikas Shivashankar, Ugur Kuter, Dana Nau, Ron Alford 981

DiscoverHistory: Understanding the Past in Planning and Execution
Matthew Molineaux, Ugur Kuter, Matthew Klenk 989

Time Bounded Adaptive A*
Carlos Hernández, Jorge Baier, Tansel Uras, Sven Koenig 997

Session 2F – Knowledge Representation & Reasoning

Memory Formation, Consolidation, and Forgetting in Learning Agents
Budhitama Subagdja, Wenwen Wang, Ah-Hwee Tan, Yuan-Sin Tan, Loo-Nin Teow 1007

Improved Use of Partial Policies for Identifying Behavioral Equivalence
Yifeng Zeng, Yinghui Pan, Hua Mao, Jian Luo 1015

Learning and Reasoning about Norms using Neural-Symbolic Systems
Guido Boella, Silvano Colombo Tosatto, Artur d'Avila Garcez, Valerio Genovese, Perotti Alan, Leendert van der Torre 1023

On Supervising Agents in Situation-Determined ConGolog
Giuseppe De Giacomo, Yves Lespérance, Christian Muise 1031

Generalized and Bounded Policy Iteration for Finitely-Nested Interactive POMDPs: Scaling Up
Ekhlas Sonu, Prashant Doshi 1039

Session 3F – Agent-based Software Development

Measuring Plan Coverage and Overlap for Agent Reasoning
John Thangarajah, Sebastian Sardina, Lin Padgham 1049

Programming Norm-Aware Agents
Natasha Alechina, Mehdi Dastani, Brian Logan 1057

Metamodel-Based Metrics for Agent-Oriented Methodologies
Noélie Bonjean, Antonio Chella, Massimo Cossentino, Marie-Pierre Gleizes, Frédéric Migeon, Valeria Seidita 1065

Comma: A Commitment-Based Business Modeling Methodology and its Empirical Evaluation
Pankaj Telang, Munindar Singh 1073

Revising Conflicting Intention Sets in BDI Agents
Steven Shapiro, Sebastian Sardina, John Thangarajah, Lawrence Cavedon, Lin Padgham 1081

Session 4F – Logics for Agency

Action models for knowledge and awareness
Hans van Ditmarsch, Tim French, Fernando R. Velázquez-Quesada 1091

Epistemic Coalition Logic: Completeness and Complexity
Thomas Ågotnes, Natasha Alechina 1099

Group Synthesis for Parametric Temporal-Epistemic Logic
Andrew Jones, Michał Knapik, Alessio Lomuscio, Wojciech Penczek 1107

A Logic of Revelation and Concealment
Wiebe van der Hoek, Petar Iliev, Michael Wooldridge 1115

State and Path Coalition Effectivity Models for Logics of Multi-Player Games
Valentin Goranko, Wojciech Jamroga 1123

Session 5F – Logic and Verification

A logic of emotions: from appraisal to coping
Mehdi Dastani, Emiliano Lorini 1133

Automatic Verification of Epistemic Specifications under Convergent Equational Theories
Ioana Boureanu, Andrew Jones, Alessio Lomuscio 1141

Semantics and Verification of Information-Based Protocols
Munindar Singh 1149

Main Program - Extended Abstracts

Innovative Applications

Emergence of Multi-generational Migration Behavior by Adaptiogenesis to Environmental Changes
Katsuya Suetsugu, Atsuko Mutoh, Shohei Kato, Hidenori Itoh 1159

A cognitive architecture for emergency response
Felipe Meneguzzi, Siddharth Mehrotra, James Tittle, Jean Oh, Nilanjan Chakraborty, Katia Sycara, Michael Lewis 1161

An Adaptive System for Proactively Supporting Sustainability Goals
Sarah Hickmott, Liam Magee, James Thom, Lin Padgham 1163

Cooperative Virtual Power Plant Formation Using Scoring Rules
Valentin Robu, Ramachandra Kota, Georgios Chalkiadakis, Alex Rogers, Nick Jennings 1165

A Storage Pricing Mechanism for Learning Agents in the Masdar City Smart Grid
Fatimah Ishowo-Oloko, Perukrishnen Vytelingum, Nick Jennings, Iyad Rahwan 1167

MAS for manufacturing control: A layered case study
Sindre Pedersen, Bjarne Foss, Ingrid Schjølberg, Johannes Tjønnås 1169

Opinion Gathering Using a Multi-Agent Systems Approach to Policy Selection
Adam Wyner, Katie Atkinson, Trevor Bench-Capon 1171

Lottery-based Resource Allocation for Plug-in Electric Vehicle Charging
Matteo Vasirani, Sascha Ossowski 1173

Virtual Agents

The Role of Social Identity, Rationality and Anticipation in Believable Agents
Rui Prada, Guilherme Raimundo, Márcia Baptista, Joana Dimas, Pedro A. Santos, Carlos Martinho, Jorge Peña, Luís Landeiro Ribeiro 1175

On-the-fly behavior coordination for interactive virtual agents - A model for learning, recognizing and reproducing hand-arm gestures online
Ulf Großekathöfer, Nils-Christian Wöhler, Thomas Hermann, Stefan Kopp 1177

Live Generation of Interactive Non-Verbal Behaviours
Ken Prepin, Catherine Pelachaud 1179

Agent Communication for Believable Human-Like Interactions between Virtual Characters
Joost van Oijen, Frank Dignum 1181

A BDI Dialogue Agent for Social Support: Specification of Verbal Support Types
Janneke van der Zwaan, Virginia Dignum, Catholijn Jonker 1183

An Agent-based Annotation Model for Narrative Media
Mario Cataldi, Rossana Damiano, Vincenzo Lombardo, Antonio Pizzo 1185

Goal-Driven Approach To Open-Ended Dialogue Management using BDI Agents
Wilson Wong, Lawrence Cavedon, John Thangarajah, Lin Padgham 1187

Distributed Punishment as a Norm-Signalling Tool
Daniel Villatoro, Giulia Andrighetto, Jordi Brandts, Jordi Sabater-Mir, Rosaria Conte 1189

The "Resource" Approach to Emotion
Sabrina Campano, Nicolas Sabouret, Etienne de Sevin, Vincent Corruble 1191

Emotional Contagion with Virtual Characters
Jason Tsai, Emma Bowring, Stacy Marsella, Milind Tambe 1193

Higher-order social cognition in rock-paper-scissors: A simulation study
Harmen de Weerd, Rineke Verbrugge, Bart Verheij 1195

Robotics

Can I trust you? Sharing information with artificial companions
Matthias Keysermann, Ruth Aylett, Sibylle Enz, Henriette Cramer, Carsten Zoll, Patricia Vargas 1197

MO-LOST: Adaptive ant trail untangling in multi-objective multi-colony robot foraging
Zhao Song, Seyed Abbas Sadat, Richard T. Vaughan 1199

Generating Strategies for Multi-Agent Pursuit-Evasion Games in Partially Observable Euclidean Space
Eric Raboin, Ugur Kuter, Dana Nau 1201

Induction and Learning of Finite-State controllers from Simulation
Matteo Leonetti, Luca Iocchi, Subramanian Ramamoorthy 1203

Spatial awareness in robotic swarms through local wireless communications
Frederick Ducatelle, Gianni Di Caro, Luca Gambardella 1205

Multi-Robot Learning by Demonstration
Michiel Blokzijl-Zanker, Yiannis Demiris 1207

Distributed Value Functions for the Coordination of Decentralized Decision Makers
Laëtitia Matignon, Laurent Jeanpierre, Abdel-Illah Mouaddib 1209

Auctioning Robotic Tasks with Overlapping Time Windows
Ernesto Nunes, Maitreyi Nanjanath, Maria Gini 1211

Real-World Testing of a Multi-Robot Team
Paul Scerri, Prasanna Velagapudi, Balajee Kannan, Abhinav Valada, Christopher Tomaszewski, John Dolan, Adrian Scerri, Kumar Shaurya Shankar, Luis Bill-Clark, George Kantor 1213

Online Planning for Large MDPs with MAXQ Decomposition
Aijun Bai, Feng Wu, Xiaoping Chen 1215

Enabling Robots to Find and Fetch Objects by Querying the Web
Thomas Kollar, Mehdi Samadi, Manuela Veloso 1217

Configurable Human-Robot Interaction for Multi-Robot Manipulation Tasks
Bennie Lewis, Gita Sukthankar 1219

Agent Reasoning

Evaluating POMDP Rewards for Active Perception
Adam Eck, Leen-Kiat Soh 1221

Finding new consequences of an observation in a system of agents
Gauvain Bourgne, Katsumi Inoue, Nicolas Maudet 1223

User-Centric Preference-Based Decision Making
Ingrid Nunes, Simon Miles, Michael Luck, Carlos de Lucena 1225

Lagrangian Relaxation for Large-Scale Multi-Agent Planning
Geoff Gordon, Pradeep Varakantham, William Yeoh, Hoong Chuin Lau, Ajay Srinivasan Aravamudhan, Shih-Fen Cheng 1227

Tree-based Pruning for Multiagent POMDPs with Delayed Communication
Frans Oliehoek, Matthijs Spaan 1229

Planning in the Logics of Communication and Change
Pere Pardo, Mehrnoosh Sadrzadeh 1231

Intention-Aware Planning under Uncertainty for Interacting with Self-Interested, Boundedly Rational Agents
Trong Nghia Hoang, Kian Hsiang Low 1233

Delayed Observation Planning in Partially Observable Domains
Pradeep Varakantham, Janusz Marecki 1235

Analysis of Methods for solving MDPs
Marek Grześ, Jesse Hoey 1237

Decentralized Multi-agent Plan Repair in Dynamic Environments
Antonín Komenda, Peter Novák, Michal Pěchouček 1239

Multimodal Trust Formation with Uninformed Cognitive Maps (UnCM)
Michele Piunti, Matteo Venanzi, Rino Falcone, Cristiano Castelfranchi 1241

Modeling Deep Strategic Reasoning by Humans in Competitive Games
Xia Qu, Prashant Doshi, Adam Goodie 1243

Coalitional Agency and Evidence-Based Ability
Nicolas Troquard 1245

Strategic voting and the logic of knowledge
Hans van Ditmarsch, Jérôme Lang, Abdallah Saffidine 1247

Exclusivity-based Allocation of Knowledge
Madalina Croitoru, Sebastian Rudolph 1249

Agent Cooperation

Role Selection in Ad Hoc Teamwork
Katie Genter, Noa Agmon, Peter Stone 1251

Integrating Self-organisation into Dynamic Coalition Formation
Dayong Ye, Minjie Zhang, Danny Sutanto 1253

An Analysis of Constructive Network Formation Models
Gary Fredericks, José Vidal 1255

On Deconflicting Local Coordination Among Agents
Manh Tung Pham, Kiam Tian Seow 1257

Hierarchical Clustering and Linguistic Mediation Rules for Multiagent Negotiation
Enrique de la Hoz, Miguel Angel Lopez Carmona, Mark Klein, Ivan Marsa-Maestre 1259

An Information Sharing Algorithm For Large Dynamic Mobile Multi-agent Teams
Linglong Zhu, Yang Xu, Paul Scerri, Han Liang 1261

Global Constraints in Distributed Constraint Satisfaction
Christian Bessiere, Ismel Brito, Patricia Gutierrez, Pedro Meseguer 1263

Multi-Agent A* for Parallel and Distributed Systems
Raz Nissim, Ronen Brafman 1265

Partial Cooperation in Multi-agent Search
Roie Zivan, Alon Grubshtein, Michal Friedman, Amnon Meisels 1267

Prioritized Shaping of Models for Solving DEC-POMDPs
Pradeep Varakantham, William Yeoh, Prasanna Velagapudi, Katia Sycara, Paul Scerri 1269

Coordinated Look-Ahead Scheduling for Real-Time Traffic Signal Control
Xiao-Feng Xie, Stephen Smith, Gregory J. Barlow 1271

Global Optimization for Multiple Agents
Brammert Ottens, Boi Faltings 1273

Scalable decentralized supply chain formation through binarized belief propagation
Toni Penya-Alba, Jesus Cerquides, Juan Antonio Rodriguez-Aguilar, Meritxell Vinyals 1275

Planning and Evaluating Multiagent Influences Under Reward Uncertainty
Stefan Witwicki, Inn-Tung Chen, Ed Durfee, Satinder Singh 1277

A Better Maximization Procedure For Online Distributed Constraint Optimization
Yoonheui Kim, Victor Lesser 1279

Agent-human Coordination with Communication Costs under Uncertainty
Asaf Frieder, Raz Lin, Sarit Kraus 1281

Token Economy for Online Exchange Systems
Jie Xu, William Zame, Mihaela van der Schaar 1283

Economic paradigms

Using the Max-Sum Algorithm for Supply Chain Formation in Dynamic Multi-Unit Environments
Michael Winsper, Maria Chli 1285

Complexity and Approximability of Social Welfare Optimization in Multiagent Resource Allocation
Nhan-Tam Nguyen, Trung Thanh Nguyen, Magnus Roos, Jörg Rothe 1287

When speed matters in learning against adversarial opponents
Mohamed Elidrisi, Maria Gini 1289

Do Experts Help in Two-Sided Search?
Yinon Nahum, David Sarne, Sanmay Das, Onn Shehory 1291

The Benefits of Search Costs in Multiagent Exploration
David Sarne, Yonatan Aumann 1293

Adaptive Negotiating Agents in Dynamic Games: Outperforming Human Behavior in Diverse Societies
Eunkyung Kim, Luyan Chi, Yu Ning, Yu-Han Chang, Rajiv Maheswaran 1295

A Robust Approach to Addressing Human Adversaries in Security Games
James Pita, Richard John, Rajiv Maheswaran, Milind Tambe, Rong Yang, Sarit Kraus 1297

Designing Better Strategies against Human Adversaries in Network Security Games
Rong Yang, Fei Fang, Albert Xin Jiang, Karthik Rajagopal, Milind Tambe, Rajiv Maheswaran . . 1299

Anytime Algorithms for Multi-agent Visibility-based Pursuit-evasion Games
Viliam Lisý, Branislav Bošanský, Michal Pěchouček 1301

Computing Optimal Security Strategies in Networked Domains: A Cost-Benefit Approach
Joshua Letchford, Yevgeniy Vorobeychik 1303

Automated Equilibrium Analysis of Repeated Games with Private Monitoring: A POMDP Approach
YongJoon Joe, Atsushi Iwasaki, Michihiro Kandori, Ichiro Obara, Makoto Yokoo 1305

Adversarial Patrolling Games
Yevgeniy Vorobeychik, Bo An, Milind Tambe 1307

Consensus Games
Julian Zappala, Natasha Alechina, Brian Logan 1309

Individual-based Stability in Hedonic Games depending on the Best or Worst Players
Haris Aziz, Paul Harrenstein, Evangelia Pyrga 1311

Influence and aggregation of preferences over combinatorial domains
Nicolas Maudet, Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable 1313

Manipulation with Randomized Tie-Breaking under Maximin
Michael Zuckerman, Jeffrey Rosenschein 1315

Learning Performance of Prediction Markets with Kelly Bettors
Alina Beygelzimer, John Langford, David Pennock 1317

TrustBets: Betting over an IOU Network
Sharad Goel, Mohammad Mahdian, David Pennock, Daniel Reeves 1319

On the Social Welfare of Mechanisms for Repeated Batch Matching
Elliot Anshelevich, Meenal Chhabra, Sanmay Das, Matthew Gerrior 1321

Merging Multiple Information Sources in Federated Sponsored Search Auctions
Sofia Ceppi, Enrico Gerding, Nicola Gatti 1323

A Truthful Learning Mechanism for Multi-Slot Sponsored Search Auctions with Externalities
Nicola Gatti, Alessandro Lazaric, Francesco Trovò 1325

Strategy-proof mechanisms for two-sided matching with minimum and maximum quotas
Suguru Ueda, Daniel Fragiadakis, Atsushi Iwasaki, Peter Troyan, Makoto Yokoo 1327

Incentives for Truthful Reporting in Crowdsourcing
Ece Kamar, Eric Horvitz 1329

Agent-based simulations

Cooperation among Malicious Agents: A General Quantitative Congestion Game Framework
Zaojie Rui, Tuanjie Fu, Darong Lai, Yichuan Jiang 1331

Opinion Convergence in Agent Networks
Sreerupa Chatterjee, Alexander Ruff, Sandip Sen 1333

Behavior Modeling From Learning Agents: Sensitivity to Objective Function Details
Robert Junges, Franziska Klügl 1335

Emergent Behaviour of Bacteria in a Multiagent System
Philip Hendrix, Elena Budrene, Benoit Morel, Igor Linkov 1337

Investigating the Role of Social Behavior in Financial Markets through Agent-Based Simulation
Alessia Mauri, Andrea Tettamanzi 1339

An Agent-Based Model for Pedestrian and Group Dynamics: Experimental and Real-World Scenarios
Giuseppe Vizzari, Lorenza Manenti 1341

The Impact of Cultural Differences on Crowd Dynamics
Natalie Fridman, Avishay Zilka, Gal Kaminka 1343

The Spanish Steps flower scam - agent-based modeling of a complex social interaction
Ladislau Bölöni 1345

Effect of defectors for cooperation: How strictly should defectors be eliminated from the newcomers?
Hitoshi Yamamoto, Isamu Okada, Yuki Ogawa 1347

Patterns of Migration and Adoption of Choices By Agents in Communities
Feyza Hafizoğlu, Sandip Sen 1349

Agent-based simulation of mobility in real-world transportation networks
Maicon Amarante, Ana Bazzan 1351

SimAnalyzer: Automated description of groups dynamics in agent-based simulations
Philippe Caillou, Javier Gil-Quijano 1353

Agent societies and Societal issues

Emergence of Cooperation through Structural Changes and Incentives in Service-Oriented MAS
Elena del Val, Miguel Rebollo, Vicent Botti 1355

Disagreement for control of rational cheating in peer review: a simulation
Mario Paolucci, Francisco Grimaldo 1357

Sub-delegation and Trust
Chris Burnett, Nir Oren 1359

A Dempster-Shafer Theory Based Witness Trustworthiness Model
Siyuan Liu, Alex C. Kot, Chunyan Miao, Yin-Leng Theng 1361

Detecting and Identifying Coalitions
Reid Kerr, Robin Cohen 1363

SARC: Subjectivity Alignment for Reputation Computation
Hui Fang, Jie Zhang, Murat Şensoy, Nadia Magnenat Thalmann 1365

The Impact of Social Placement of Non-Learning Agents on Convention Emergence
Nathan Griffiths, Sarabjot Singh Anand 1367

Handling Change in Normative Specifications
Duangtida Athakravi, Domenico Corapi, Alessandra Russo, Marina De Vos, Julian Padget, Ken Satoh 1369

A Context-aware Normative Structure in MAS
Jie Jiang, Huib Aldewereld, Virginia Dignum, Yao-Hua Tan 1371

A Programming Approach to Monitoring Communication in an Organisational Environment
Mehdi Dastani, Leendert van der Torre, Neil Yorke-Smith 1373

On modeling punishment in multi-agent systems
Subhasis Thakur, Guido Governatori, Abdul Sattar 1375

Strategic Pseudonym Change in Agent-Based E-Commerce
José Such, Emilio Serrano, Vicent Botti, Ana García-Fornes 1377

Multi-dimensional Transition Deliberation for Organization Adaptation in Multiagent Systems
Juan M. Alberola, Vicente Julian, Ana García-Fornes 1379

Using a hierarchy of coordinators to overcome the frontier effect in social learning
Sherief Abdallah 1381

Learning and Adaptation

Towards Student/Teacher Learning in Sequential Decision Tasks
Lisa Torrey, Matthew Taylor 1383

Bayes-Optimal Reinforcement Learning for Discrete Uncertainty Domains
Emma Brunskill 1385

Algorithms for Scaling in a General Episodic Memory
Nate Derbinsky, Justin Li, John Laird 1387

Break with agents who listen to too many others (at least when making Boolean decisions!)
Daniel Epstein, Ana Bazzan, André Machado 1389

Adaptive Agents on Evolving Networks
Ardeshir Kianercy, Aram Galstyan, Armen Allahverdyan 1391

A Common Gradient in Multi-agent Reinforcement Learning
Michael Kaisers, Daan Bloembergen, Karl Tuyls 1393

Combining Independent and Joint Learning: a Negotiation based Approach
Reinaldo Bianchi, Ana Bazzan 1395

Modeling Difference Rewards for Multiagent Learning
Scott Proper, Kagan Tumer 1397

Revenue prediction in budget-constrained sequential auctions with complementarities
Sicco Verwer, Yingqian Zhang 1399

An RL approach to Common-Interest Continuous Action Games
Abdel Rodríguez, Peter Vrancx, Ricardo Grau, Ann Nowé 1401

Agreement Technologies

Selecting judgment aggregation rules for NAO robots: an experimental approach
Vijayalakshmi Ganesan, Marija Slavkovik, Sergio Sousa, Leendert van der Torre 1403

Distance-based Rules for Weighted Judgment Aggregation
Marija Slavkovik, Wojciech Jamroga 1405

Bribery in Voting Over Combinatorial Domains is Easy
Nicholas Mattei, Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable 1407

On the benefits of argumentation schemes in deliberative dialogue
Alice Toniolo, Timothy Norman, Katia Sycara . . . 1409

Testing the Benefits of Structured Argumentation in Multi-Agent Deliberation Dialogues
Eric Kok, John-Jules Meyer, Henry Prakken, Gerard Vreeswijk . . . 1411

Knowing Each Other in Argumentation-based Negotiation
Elise Bonzon, Yannis Dimopoulos, Pavlos Moraitis . . . 1413

Branch and Bound for Negotiations in Large Agreement Spaces
Dave de Jonge, Carles Sierra . . . 1415

Collaborative Job Processing on a Single Machine - A Multi-Agent Weighted Tardiness Problem
Fabian Lang, Andreas Fink . . . 1417

Determining the Willingness to Comply With Norms
Natalia Criado, Estefanía Argente, Pablo Noriega, Vicent Botti . . . 1419

The Dutch eat at 5:30 pm: Shared Strategies for Agent Reasoning
Amineh Ghorbani, Huib Aldewereld, Virginia Dignum, Pablo Noriega . . . 1421

Specifying and reasoning about normative systems in deontic logic programming
Ricardo Gonçalves, José Alferes . . . 1423

Normative Systems require Hybrid Knowledge Bases
Marco Alberti, Ana Sofia Gomes, Ricardo Gonçalves, Matthias Knorr, João Leite, Martin Slota . 1425

Systems and Organisation

Self-management of Ambient Intelligence Systems: a Pure Agent-based Approach
Inmaculada Ayala, Mercedes Amor, Lidia Fuentes . . . 1427

Enhancing Decentralized Service Discovery through Structural Self-Organization
Elena del Val, Matteo Vasirani, Miguel Rebollo, Alberto Fernandez . . . 1429

Cloning, Resource Exchange and Relation Adaptation: A Self-organising Multi-Agent Framework
Dayong Ye, Minjie Zhang, Danny Sutanto . . . 1431

Agent-based system development

Dynamic change impact analysis for maintaining and evolving agent systems
Hoa Dam, Aditya Ghose . . . 1433

Supporting User-Centric Business Processes with WADE
Federico Bergenti, Giovanni Caire, Danilo Gotta . . . 1435

OrgMAP: An Organization-based Approach for Multi-Agent Programming
Cuiyun Hu, Xinjun Mao, Yin Chen, Huiping Zhou . . . 1437

MAPLE: Multi-Agent Programming with Letter Exchanges on Sensor Networks
Tiffany Yi-Ting Tsao, Wan-rong Jih, Jane Yung-jen Hsu . . . 1439

Agent theories - Models and Architectures

Efficient Context Free Parsing of Multi-agent Activities for Team and Plan Recognition
Bikramjit Banerjee, Jeremy Lyle, Landon Kraemer . . . 1441

Agent Deliberation via Forward and Backward chaining in Linear Logic
Luke Trodd, James Harland, John Thangarajah . . . 1443

On the Failure of Game Theoretic Approach for Distributed Deadlock Resolution
Nadav Sofy, David Sarne . . . 1445

Bounded Model Checking for Knowledge and Linear Time
Artur Męski, Wojciech Penczek, Bozena Woźna-Szcześniak, Maciej Szreter, Andrzej Zbrzezny . . 1447

The role of identity in agent design
Ines Di Loreto, Fabien Hervouet . . . 1449

Demonstrations

SAFEPED: Agent-Based Environment for Estimating Accident Risks at the Road Black Spots
Gennady Waizman, Itzhak Benenson 1453

Sustainable Multiagent Application to Conserve Energy
Jun-young Kwak, Pradeep Varakantham, Rajiv Maheswaran, Milind Tambe, Farrokh Jazizadeh, Geoffrey Kavulya, Laura Klein, Burcin Becerik-Gerber, Timothy Hayes, Wendy Wood 1455

Migrating Artificial Companions
Iain Wallace, Michael Kriegel, Ruth Aylett 1457

Effective Methods for Generating Collision Free Paths for Multiple Robots based on Collision Type
Fan Liu, Ajit Narayanan, Quan Bai 1459

Decentralised stable coalition formation among energy consumers in the smart grid
Filippo Bistaffa, Alessandro Farinelli, Meritxell Vinyals, Alex Rogers 1461

Learning to be Scientists via a Virtual Field Trip
Deborah Richards, Michael J. Jacobson, Meredith Taylor, Anne Newstead, Charlotte Taylor, John Porte, Iwan Kelaiah, Nader Hanna 1463

Virtual Characters in Agent-Augmented Co-Space
Yi-Lin Kang, Budhitama Subagdja, Ah-Hwee Tan, Yew-Soon Ong, Chunyan Miao 1465

ARGUS: A Coordination System to Provide First Responders with Live Aerial Imagery of the Scene of a Disaster
Francesco Maria Delle Fave, Alex Rogers, Nick Jennings 1467

Pogamut Toolkit
Jakub Gemrot, Michal Bída, Cyril Brom 1469

An Intelligent Agent for Home Heating Management
Alex Rogers, Sasan Maleki, Siddhartha Ghosh, Nick Jennings 1471

Tactical Operations of Multi-Robot Teams in Urban Warfare
Peter Novák, Antonín Komenda, Viliam Lisý, Branislav Bošanský, Michal Čáp, Michal Pěchouček 1473

MITRO: an augmented mobile telepresence robot with assisted control
Sjriek Alers, Daan Bloembergen, Max Bügler, Daniel Hennes, Karl Tuyls 1475

Toolkit for Teaching Steering Behaviors for 3D Human-like Virtual Agents
Markéta Popelová, Cyril Brom, Jakub Tomek, Michal Bída 1477

A Development Environment for Engineering Intelligent Avatars for Semantically-enhanced Simulated Realities
Stefan Warwas, Matthias Klusch, Klaus Fischer, Philipp Slusallek 1479

Running Experiments on DipGame Testbed
Angela Fabregues, Santiago Biec, Carles Sierra 1481

v-mWater: a 3D Virtual Market for Water Rights
Pablo Almajano, Tomas Trescak, Marc Esteva, Inmaculada Rodriguez, Maite Lopez-Sanchez . . . 1483

Context-Aware MAS to Support Elderly People
Boštjan Kaluža, Mitja Luštrek, Erik Dovgan, Matjaž Gams 1485

Agent Based Monitoring of Gestational Diabetes Mellitus
René Schumann, Stefano Bromuri, Johannes Krampf, Michael Schumacher 1487

Protos: A Cross-Organizational Business Modeling Tool
Anup Kalia, Pankaj Telang, Munindar Singh 1489

Expectation and Complex Event Handling in BDI-based Intelligent Virtual Agents
Surangika Ranathunga, Stephen Cranefield 1491

ARGOS: Simulating Migration Processes
Oscar Alvarado, N. Ruiz, Adriana Giret, Vicente Julian, Vicent Botti, Victor Perez, Rosa Maria Rodriguez 1493

CALU: Collision Avoidance with Localization Uncertainty
Daniel Claes, Daniel Hennes, Karl Tuyls, Wim Meeussen 1495

Stigmergic Coverage Algorithm for Multi-Robot Systems
Bijan Ranjbar-Sahraei, Gerhard Weiss, Ali Nakisaee 1497

Infraworld, a Multi-agent Based Framework to Assist in Civil Infrastructure Collaborative Design
Jaume Faus, Francisco Grimaldo 1499

AgentPolis: Towards a Platform for Fully Agent-based Modeling of Multi-Modal Transportation
Michal Jakob, Zbyněk Moler, Antonín Komenda, Zhengyu Yin, Albert Xin Jiang, Matthew Johnson, Michal Pěchouček, Milind Tambe 1501

Distributed Consensus for Interaction between Humans and Mobile Robot Swarms
Alessandro Giusti, Jawad Nagi, Luca Gambardella, Gianni Di Caro 1503

Team-It: Location-Based Mobile Games for Multi-Agent Coordination and Negotiation
Spencer Frazier, Yu-Han Chang, Alex Newnan, Rajiv Maheswaran 1505

GaTAC: A Scalable and Realistic Testbed for Multiagent Decision Making
Ekhlas Sonu, Prashant Doshi 1507

Session 1D
Social Choice I

Strategyproof Approximations of Distance Rationalizable Voting Rules[*]

Travis C. Service
Department of Electrical Engineering and Computer Science
Vanderbilt University
travis.c.service@vanderbilt.edu

Julie A. Adams
Department of Electrical Engineering and Computer Science
Vanderbilt University
juile.a.adams@vanderbilt.edu

ABSTRACT

This paper considers randomized strategyproof approximations to distance rationalizable voting rules. It is shown that the *Random Dictator* voting rule (return the top choice of a random voter) nontrivially approximates a large class of distances with respect to unanimity. Any randomized voting rule that deviates too greatly from the Random Dictator voting rule is shown to obtain a trivial approximation (i.e., equivalent to ignoring the voters' votes and selecting an alternative uniformly at random).

The outlook for consensus classes, other than unanimity is bleaker. This paper shows that for a large number of distance rationalizations, with respect to the majority and Condorcet consensus classes that no strategyproof randomized rule can asymptotically outperform uniform random selection of an alternative. This paper also shows that veto cannot be approximated nontrivially when approximations are measured with respect to minimizing the number of vetoes an alternative receives.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Theory

Keywords

voting, distance rationalization, strategyproof, approximation

1. INTRODUCTION

The Gibbard-Satterthwaite theorem [11, 16] states that any natural voting procedure can be manipulated. A growing body of work in computation social choice has investigated methods for circumventing the Gibbard-Satterthwaite theorem through hardness of computation [2, 3, 8]. For example, the manipulation problem for many common voting rules has been shown to be NP-hard. However, these are worst case results and say nothing about the manipulation problem on average. Conitzer and Sandholm [3] show that many voting rules can be manipulated in polynomial time for a large fraction of elections.

Recently, Procaccia [15] considered approximating score-based voting rules by strategyproof randomized voting rules. However, Procaccia's approach is limited to rules that have a natural measure of score. This paper studies the approximation of common voting rules with respect to the distance rationalization framework. Approaching the approximation of voting rules from the viewpoint of distance rationalization permits the approximation of voting rules that do not necessarily have a natural measure of score.

In many elections there is an alternative that is a clear winner. For example, if every voter prefers alternative w to every other alternative, then w is the clear winner. Similarly, if w is the Condorcet winner (i.e., w is preferred to every other alternative by a majority of the voters), then w is the clear winner of the election. Both of the previous examples are different notions of consensus in an election. In the second example, a consensus is said to exist in an election whenever there is a Condorcet winner. The Condorcet consensus class is the subset of elections in which their exists a Condorcet winner. The distance rationality framework casts voting in the context of selecting an alternative that is closest to being a consensus winner. The notion of closeness in the distance rationalization framework is formalized by employing distances over elections. The distances employed provide a natural means by which to measure the approximation ratio obtained by strategyproof randomized voting rules.

This paper shows that under the distance rationalizability framework, consensus with respect to Unanimity and arbitrary votewise distances with l_p norms can be approximated to a nontrivial factor[1]. Thus, all positional scoring rules and their (pseudo-)distance rationalizations, given by Elkind et al. [4] can be nontrivially approximated. Similarly, $2-\frac{2}{n}$ and $O(m)$ approximations are obtained for the standard distance rationalizations of plurality and Borda voting, respectively. These approximation ratios are significantly better than random selection of an alternative, which results in a $\Omega(n)$ and $\Omega(nm)$ approximation for plurality and Borda, respectively.

Surprisingly, the Random Dictator rule (select the first

[*]This research has been supported by an ONR DEPSCOR Award # N000140911161.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

[1]An approximation ratio is said to be trivial if it is achieved by selecting an alternative uniformly at random.

choice alternative of a random voter) is shown to nontrivially approximate all votewise distance rationalizations with respect to Unanimity and l_p norm. It is shown that deviating too much from the Random Dictator rule results in a trivial ratio under the l_1 norm.

Lower bounds are provided for a number of distance rationalizations. For example, a bound of $\Omega(m)$ is proven for approximating the standard distance rationalization of Borda.

Approximation ratios obtained for a given voting rule are highly dependent on the distance rationalization considered. For example, under unanimity and the discrete distance, plurality can be approximated to a factor of $2 - \frac{2}{n}$. However, under the same distance and the majority consensus class, plurality cannot be approximated better than $\Omega(n)$.

The outcome for consensus classes, other than unanimity is bleaker. This paper shows for a number of other distance rationalizable voting rules that essentially one cannot do better than uniform random selection of an alternative.

The remainder of this paper is as follows. Section 2 presents preliminary definitions and related work. Section 3 presents a number of upper and lower bounds on the approximations obtainable by randomized strategyproof voting rules for a number of distance rationalizations. Section 4 concludes with some final remarks.

2. PRELIMINARIES

This section begins by presenting the basic notation and definitions employed throughout this paper and concludes by discussing related work.

2.1 Elections and Strategy-Proofness

An election $E = (A, V)$ consists of a set of alternatives $A = \{a_1, \cdots, a_m\}$ and a tuple of voters $V = \{v_1, \cdots, v_n\}$. Each voter v_i has a strict total preference order $\succ_i$ over the set of alternatives A. Let $v \in V$ and $a \in A$. Define $v(a)$ to be the rank of alternative a in v's preference order. A tuple of preference orders $(\succ_1, \cdots, \succ_n)$ is referred to as a preference profile. A voting rule f is a function that maps each preference profile to a winning alternative.

Informally, a voting rule is manipulable if there exists a preference profile under which some voters can benefit (possibly in expectation) by misrepresenting their true preferences. A voting rule is strategyproof if it is not manipulable. That is, f is manipulable if there exists a preference profile $P = (\succ_1, \cdots, \succ_n)$, a voter v_i and a preference order $\succ'_i$ such that $f(P') \succ_i f(P)$ where $P' = (\succ_1, \cdots, \succ'_i, \cdots, \succ_n)$. If f is a randomized voting rule, then f is manipulable if a voter can increase its expected utility by misrepresenting her preferences under some preference profile.

The following two classes of deterministic voting rules are used throughout this paper.

Definition 1. *A deterministic voting rule is said to be unilateral if it is a function of exactly one voters' vote.*

Definition 2. *A deterministic voting rule is said to be duple if it always elects one of two fixed alternatives.*

The following result provides necessary conditions for a randomized voting rule to be strategyproof.

Theorem 1 ([12]). *If R is a strategyproof randomized voting rule, then R is a probability distribution over unilateral and duple rules.*

2.2 Distance Rationalization

A consensus in an election $E = (A, V)$ is a clear winner[2] $w \in A$. For example, if every voter in V ranks alternative w first, then w can be considered the consensus winner. Formally, a consensus class $\mathcal{K} = (\mathcal{E}, \mathcal{W})$ is a tuple, where $\mathcal{E}$ is a set of elections and $\mathcal{W} : \mathcal{E} \to A$ is a function that determines for each election $E \in \mathcal{E}$ the consensus winner. We consider three consensus classes in which there exists a clear winner [4, 5, 6, 13, 14].

1. Unanimity ($\mathcal{U}$): Consists of all elections in which every voter ranks the same alternative first. The consensus winner is the alternative preferred by all voters.
2. Majority ($\mathcal{M}$): Consists of all elections in which some alternative is ranked first by a strict majority of the voters. The consensus winner is the unique alternative that more than half of the voters rank first.
3. Condorcet ($\mathcal{C}$): Consists of all elections in which there exists a Condorcet winner (i.e., an alternative that defeats every other alternative in a pairwise election). The Condorcet winner is the consensus winner.

A consensus class can be extended to a voting rule over arbitrary elections by defining the winning alternative in an election E to be the alternative that is closest to being a consensus winner. Such an extension requires a notion of distance between elections.

Informally, a distance function $d : X \times X \to \mathbb{R} \cup \{\infty\}$ on a set X is a function mapping pairs of elements in X to a non-negative real value representing the distance between the elements. A pseudo-distance function is similar to a distance function but there may be distance 0 between two distinct members of X. We are interested in (pseudo-)distances over the set of elections.

Definition 3. *Let $\mathcal{K} = (\mathcal{E}, \mathcal{W})$ be a consensus class and d a (pseudo-)distance function on the set of elections. A voting rule, f, is $(\mathcal{K}, d)$-rationalizable if for every election E*

$$f(E) \in \operatorname*{argmin}_{a \in A} \left\{ \min_{E' \in \mathcal{E} \,:\, a \in \mathcal{W}(E')} \{d(E, E')\} \right\}.$$

Refer to value $d(a) = \min_{E' \in \mathcal{E} \,:\, a \in \mathcal{W}(E')} \{d(E, E')\}$ as the distance score a. Distance rationalizable voting rules select an alternative with the least distance score. In general, there may be multiple alternatives that tie for the best distance score. Voting rules break ties in such situations.

Some of the results in this paper apply to arbitrary distances of a particular form. Let d be a distance over preference orders and let N be a norm on $\mathbb{R}^n$. Then the function

$$\hat{d}(E, E') := \begin{cases} \infty & \text{if } A \neq A' \text{ or } |V| \neq |V'| \\ N(d(v_1, v'_1), \cdots, d(v_n, v'_n)) & \text{otherwise} \end{cases}$$

is a distance over preference profiles. Distances of the above form are referred to as votewise distances [5].

Many of the norms used in votewise distances in the literature (and in the results presented in this paper) are the l_p norms ($p \in \mathbb{N} \cup \{\infty\}$) defined for each $r_1, \cdots, r_n \in \mathbb{R}$ as

$$l_p(r_1, \cdots, r_n) := \begin{cases} (r_1^p + \cdots + r_n^p)^{\frac{1}{p}} & \text{if } p \in \mathbb{N} \\ \max\{r_1, \cdots, r_n\} & \text{if } p = \infty. \end{cases}$$

[2]Sometimes multiple consensus winners are allowed.

Unless otherwise stated, it is assumed that the norm N in the definition of $\hat{d}$ is the l_1 norm (i.e., the sum of the distances over individual votes). For any norm N and any distance over preference orders d, define $N \circ d$ to be the corresponding votewise distance.

Several common votewise distances are employed throughout this paper.

1. $d_{swap}(v, v')$ represents the number of adjacent pairs that must be swapped in the preference order v in order to obtain v'.

2. $d_{disc}(v, v')$ is 0 if $v = v'$ and 1 otherwise.

3. Fix m numbers $\alpha_1 \geq \cdots \geq \alpha_m$. Define the distance: $d_\alpha(v, v') = \sum_{a \in A} |\alpha_{v(a)} - \alpha_{v'(a)}|$.

We also consider the following non-votewise distance, d_{ins}. d_{ins} is defined as follows. Let $E = (A, V)$ and $E' = (A, V')$ be elections. For each $v_i \in V$ let $\succ_{v_i}$ be v_i's preference order. Likewise, for $v_i \in V'$, let $\succ'_{v_i}$ be v_i's preference order. Then $d_{ins}(E, E') = |V \setminus V'| + |V' \setminus V| + 2|\{v_i \in V \cap V' : \succ_{v_i} \neq \succ'_{v_i}\}|$. Elkind et al. [7] show that d_{ins} under $\mathcal{C}$ is essentially equivalent to considering the number of voters that need be added to an election to make a given candidate the Condorcet winner. More formally, Elkind et al. show that given an election $E = (A, V)$ and an alternative $a \in A$, there exists an election $E_1 = (A, V \cup V_1)$ such that a is the Condorcet winner and $|V_1| \leq k$, if and only if there exists an election $E_2 = (A, V_2)$ such that $d_{ins}(E, E_2) \leq k$.

2.3 Voting Rules

Several common voting rules considered in this paper are now defined. The employed voting rules assign alternatives a score based upon the voters' preferences and then select the alternative with the best score. We present the definitions of the voting rules for situations where no ties in score occur. When multiple alternatives tie for the best score, an arbitrary, but fixed, tie breaking scheme is employed to select a single alternative. All of the presented results are unaffected by the actual tie breaking scheme employed.

1. Positional Scoring Rules: Let $\alpha = (\alpha_1, \cdots, \alpha_m)$ be a vector of m integers with $\alpha_1 \geq \cdots \geq \alpha_m$. Under the voting rule R_α, alternative a is awarded α_k points for each voter that ranks a in position k. The winner under R_α is the alternative with the largest score. Plurality is defined by $\alpha = (1, 0, \cdots, 0)$, Borda by $\alpha = (m-1, m-2, \cdots, 1, 0)$ and veto by $\alpha = (1, 1, \cdots, 1, 0)$.

   Elkind et al. [4] show that every positional scoring rule R_α is pseudo-distance rationalizable with respect to $(\mathcal{U}, \hat{d}_\alpha)$. Further, plurality and Borda are rationalizable with respect to $\mathcal{U}$ with the distances $\hat{d}_{disc}$ and $\hat{d}_{swap}$, respectively.

2. Maximin: The Maximin score of an alternative a in an election E is

$$sc(a) = \min_{b \in A \setminus \{a\}} \{|\{v_i \in V : a \succ_i b\}|\}.$$

   That is, the Maximin winner of an election E is the alternative that performs the best in their worst pairwise election against any other alternative. Maximin is $(\mathcal{C}, d_{ins})$-rationalizable [4].

3. Dodgson: The Dodgson score of an alternative a in an election $E = (V, A)$ is the minimum number of swaps of adjacent alternatives in the preferences of voters to make a the Condorcet winner. The winner is the alternative with the smallest Dodgson score. Dogdson is $(\mathcal{C}, \hat{d}_{swap})$-rationalizable.

2.4 Related Work

The manipulation problem has received much attention [1, 10, 15]. For a recent survey, see Faliszewski and Procaccia [9].

Recent work has investigated the approximation of common voting rules by randomized voting rules that are strategyproof (or almost strategyproof). Procaccia [15] recently quantified the level of approximation that can be obtained for a number of score based voting rules.

Birell and Pass [1] relax the requirement that the randomized approximation be strategyproof. Rather, Birell and Pass consider randomized voting rules such that no voter can improve their expected utility by more than ϵ by voting untruthfully. On a positive note, Birell and Pass show that for ϵ sufficiently large $(\omega(\frac{1}{n}))$, every deterministic voting rule can be approximated by an ϵ-strategyproof randomized rule. However, for $\epsilon = o(\frac{1}{n})$, every ϵ-strategyproof voting rule is a distribution over unilateral and duple rules (just as strategyproof randomized voting rules are).

One drawback of Procaccia's approach is that it is limited to score based voting rules. Birell and Pass [1] measure the quality of an approximation by the minimum number of votes that must be changed in order to elect the approximate winner. Hence, Birell and Pass' results apply to arbitrary voting rules. This paper investigates the use of the distance rationalization framework to measure the quality of approximations obtained by randomized strategyproof voting rules. Like Birell and Pass' approach, our approach allows for defining approximations to non-score based rules.

3. RESULTS

All of the results in this paper are stated for particular distance rationalizations. For example, rather than saying plurality can be approximated well, we state the distance rationalization by which we are approximating. It will be shown that the same voting rule can be approximated to different degrees, depending upon the rationalization used. Hence, it is meaningless to say, in this framework, that, for example, plurality can be approximated well, since it depends on which rationalization of plurality is employed.

Let $(\mathcal{K}, d)$ be a distance rationalization. Let E be an election and let w be the winning alternative under $(\mathcal{K}, d)$. A natural measure of the approximation ratio of a strategyproof randomized voting rule R with respect to $(\mathcal{K}, d)$ is $\mathbb{E}\left(\frac{d(R(E))}{d(w)}\right)$, with the understanding that $\frac{0}{0} = 1$ and $\frac{n}{0} = +\infty$ for any $n > 0$. It is straightforward to observe that for the Majority and Condorcet consensus classes that no strategyproof voting rule is guaranteed to return a consensus winner when one exists. Hence, the approximation ratio obtained by any strategyproof voting rule to distance rationalizations with respect to Majority or Condorcet is $+\infty$. However, simply stating an approximation ratio of $+\infty$ for these consensus classes is less than desirable as it may be the case that, for example, when there is a majority winner, the given strategyproof voting rule always selects an alternative that is close to being a consensus winner. Therefore, our lower bound results consider the approximation ratio obtained by randomized strategyproof voting rules on elections where there is no consensus winner.

The presented upper bounds, with respect to $\mathcal{U}$ all employ a strategyproof randomized voting rule that is consistent with $\mathcal{U}$. A unanimous winner, if it exists, is always selected. Hence, if E is a member of the consensus class, then the result is a perfect approximation.

3.1 Upper Bounds

Surprisingly the following strategyproof randomized voting rule obtains a nontrivial approximation ratio with respect to $(\mathcal{U}, \hat{d})$, when d is any votewise distance and $\hat{d} = l_p \circ d$.

Random Dictator. *Uniformly at random, select a voter v. Return v's first choice.*

Theorem 2 shows that Random Dictator obtains a good approximation to many distance rationalizations.

Theorem 2. *Let d be a distance over preference orders, $p \in \mathbb{N} \cup \{\infty\}$, and let $\hat{d} = l_p \circ d$ be the corresponding distance over elections. For preferences over m alternatives and $x \in A$, let $d_{Max}(m, x)$ be the maximum distance between any preference order $\succ$, that does not rank x first, to the closest preference order to $\succ$ that does rank x first. Let $d_{Max}(m) = \max_{x \in A}\{d_{Max}(m, x)\}$. Let $d_{Min}(m, x)$ be the minimum distance between any preference order $\succ$, that does not rank x first, to any preference order that does rank x first. Let $d_{Min}(m) = \min_{x \in A}\{d_{Min}(m, x)\}$.*

Random Dictator approximates $(\mathcal{U}, \hat{d})$ to within a factor of

$$\left(1 - \frac{1}{n}\right) \cdot \left(\frac{d_{Max}(m)}{d_{Min}(m)} + 1\right).$$

Proof. Let $E = (A, V)$ be an election with n voters and m alternatives. Let $w \in A$ be a winning alternative in $(\mathcal{U}, \hat{d})$ and let x be the number of voters that rank w first. If w is the unanimous winner, then Random Dictator selects w and obtains a perfect approximation. So assume that $x \leq n - 1$.

For $a \in A$, let p_a be the probability that a is selected by Random Dictator. That is, p_a is the ratio of the number of voters that rank a first to the total number of voters n. Hence, $p_w = \frac{x}{n}$.

Note that if an alternative $a \in A \setminus \{w\}$ is selected by Random Dictator, then a must be ranked first by at least one voter. If $p = \infty$, then the maximum possible distance score of any candidate is $d_{Max}(m)$ and the minimum possible distance score of w is $d_{Min}(m)$. Thus, the expected distance score is

$$\begin{aligned}
\mathbb{E}\left(\frac{\hat{d}(R(E))}{\hat{d}(w)}\right) &\leq \frac{x}{n} + \frac{n-x}{n} \cdot \frac{d_{Max}(m)}{d_{Min}(m)} \\
&\leq \left(1 - \frac{1}{n}\right)\left(\frac{d_{Max}(m)}{d_{Min}(m)} + 1\right).
\end{aligned}$$

Similarly, if $p \in \mathbb{N}$, then the maximum distance score of a possible winner under Random Dictator is $(n-1)^{\frac{1}{p}} d_{Max}(m)$. Likewise, the minimum possible distance score of w is $(n - x)^{\frac{1}{p}} d_{Min}(m)$. Thus, the approximation ratio obtained by Random Dictator is

$$\begin{aligned}
\mathbb{E}\left(\frac{\hat{d}(R(\succ))}{\hat{d}(w)}\right) &= \frac{1}{\hat{d}(w)}\left[\sum_{a \in A} p_a \hat{d}(a)\right] \\
&= \frac{1}{\hat{d}(w)}\left[\frac{x}{n}\hat{d}(w) + \sum_{a \in A \setminus \{w\}} p_a \hat{d}(a)\right] \\
&\leq \frac{x}{n} + \frac{1}{\hat{d}(w)} \sum_{a \in A \setminus \{w\}} p_a (n-1)^{\frac{1}{p}} d_{Max}(m) \\
&\leq \frac{x}{n} + \frac{n-x}{n} \cdot \frac{(n-1)^{\frac{1}{p}} d_{Max}(m)}{(n-x)^{\frac{1}{p}} d_{Min}(m)} \\
&= \frac{x}{n} + \frac{(n-x)^{1-\frac{1}{p}}}{n} \cdot \frac{(n-1)^{\frac{1}{p}} d_{Max}(m)}{d_{Min}(m)} \\
&\leq \frac{n-1}{n} + \frac{(n-1)^{1-\frac{1}{p}}}{n} \cdot \frac{(n-1)^{\frac{1}{p}} d_{Max}(m)}{d_{Min}(m)} \\
&\leq \left(1 - \frac{1}{n}\right)\left(\frac{d_{Max}(m)}{d_{Min}(m)} + 1\right).
\end{aligned}$$

□

Recall that every positional scoring rule is pseudo-distance rationalizable with respect to unanimity under the votewise distance $\hat{d}_\alpha = l_1 \circ d_\alpha$ [4]. Note that if $\alpha_i \neq \alpha_j$ whenever $i \neq j$, then $d_{Min}(m) = 2(\alpha_1 - \alpha_2)$ and $d_{Max}(m) = 2(\alpha_1 - \alpha_m)$. Corollary 1 shows that all positional scoring rules with $\alpha_i \neq \alpha_j$ are approximated by Random Dictator to within a nontrivial factor.

Corollary 1. *If R_α is a positional scoring rule, such that $\alpha_i \neq \alpha_j$ whenever $i \neq j$, then Random Dictator approximates $(\mathcal{U}, \hat{d}_\alpha)$ to within a factor of*

$$\left(1 - \frac{1}{n}\right) \cdot \left(\frac{\alpha_1 - \alpha_m}{\alpha_1 - \alpha_2} + 1\right).$$

If $\alpha_1 = \alpha_2 \neq \alpha_3$, then Random Dictator does not approximate $(\mathcal{U}, \hat{d}_\alpha)$ well. Consider the election in which each voter ranks alternative w second and no other alternative is ranked first more than once. The distance score of w is 0, but the distance score of every other alternative is at least $(n-1)(\alpha_1 - \alpha_3)$. However, Random Dictator never selects w in such elections.

Theorem 2 allows one to obtain nontrivial approximation ratios with respect to the standard distance rationalizations of common voting rules. For example, Plurality is known to be $(\mathcal{U}, \hat{d}_{disc})$-rationalizable. Since, under $\hat{d}_{disc}$, $d_{Min}(m) = d_{Max}(m) = 1$, Random Dictator approximates $(\mathcal{U}, \hat{d}_{disc})$ to within a factor of $2 - \frac{2}{n}$. Hence, Random Dictator is significantly better than uniform random selection of a candidate, which obtains an approximation ratio of $\Omega(n)$ in the worst case (e.g., when some $w \in A$ is ranked first by all but one voter).

Similarly, Borda is $(\mathcal{U}, \hat{d}_{swap})$-rationalizable. Notice that in this case $d_{Max}(m) = \Theta(m)$ and $d_{Min}(m) = 1$. Hence, Random Dictator obtains a $O(m)$ approximation. Note that uniformly at random selecting an alternative obtains an approximation ratio of $\Theta(nm)$ to $(\mathcal{U}, \hat{d}_{swap})$ (e.g., when one alternative is ranked first by all but one voter and every other alternative obtains the same distance score of $\Theta(nm)$).

It may be expected that the voting rules presented by Procaccia [15] will outperform Random Dictator, since an alternative is selected with probability proportional to its score. However, electing an alternative with probability proportional to its Borda score obtains a $\Theta(nm)$ approximation to $(\mathcal{U}, \hat{d}_{swap})$. Consider the election in which each voter has the same preference order. Let A' be the set of alternatives ranked in the lower $\frac{m}{2}$ positions by each voter. The sum of the scores of alternatives in A' is $\Omega(nm^2)$. Thus, the probability that some alternative in A' is selected is $\Theta(1)$. Therefore, selecting an alternative with probability proportional to its Borda score results in a $\Omega(nm)$ approximation. In the case of Plurality, the strategyproof randomized voting rule given by Procaccia [15] is the Random Dictator rule.

3.2 Lower Bounds

All of the presented lower bounds employ Yao's Minimax principle [17]. Consider the following two player game. The space of the first player's strategies consist of all duple and unilateral voting rules and the second player's strategies consist of all preference profiles. Given a choice of a pure strategy for each player, the outcome is defined to be the approximation ratio obtained by the unilateral or duple rule selected by the first player on the preference profile selected by the second player. Since the first player's pure strategies consist of all unilateral and duple rules, the first player's mixed strategies contain all strategyproof randomized voting rules.

Let P be any probability distribution over preference profiles. In this setting, Yao's Minimax principle states that the approximation ratio obtained by any strategyproof randomized rule is at most the approximation ratio obtained by the best deterministic duple or unilateral rule over P. Hence, the performance of any strategyproof randomized voting rule can be lower bounded by constructing a probability distribution over preference profiles on which no unilateral or duple rule performs well in expectation.

When employing Yao's principle for proving lower bounds, there are two cases to be considered: unilateral rules and duple rules. For our purposes, it will suffice to treat a duple rule as a set of two alternatives. Hence, for a duple rule D, we may treat D as a set of two alternatives as we are indifferent to how D selects a winner.

Let $\hat{d} = l_1 \circ d$ be a votewise distance. Let $\succ$ be a preference profile over m alternatives and let $a \in A$. Let $d_a(\succ)$ be the minimum distance between $\succ$ and any other preference profile $\succ'$ that ranks a first. We say that d and $(\mathcal{U}, \hat{d})$ are *rank based* if $d_a(\succ)$ depends only on the rank of a in $\succ$. Thus, $d_a(\succ)$ is independent of how the alternatives in $A \backslash \{a\}$ are ranked and is also independent of the alternative a. That is, if $\succ_a$ is any preference order that ranks a in position k and $\succ_b$ is any preference order that ranks b in position k, then $d_a(\succ_a) = d_b(\succ_b)$. Hence, there exists a function $r_d : \mathbb{N} \to \mathbb{R}^{\geq 0}$ such that $d_a(\succ) = r_d(k)$, where k is the rank of a in $\succ$. Note that all the votewise distances defined in this paper are rank based. Define $d_{Ave}(m) = \frac{1}{m}\sum_{k=1}^{m} r_d(k)$.

The Random Dictator rule is not very desirable as it ignores the preferences of all but one randomly selected voter. However, as the next result shows, it is not possible to deviate too greatly from the Random Dictator rule and still obtain a nontrivial approximation to $(\mathcal{U}, \hat{d})$, where $\hat{d}$ is any rank based votewise distance.

Theorem 3. *Let $(\mathcal{U}, \hat{d})$ be rank based and let R be a strategyproof randomized voting rule, such that with probability p, R selects a duple rule and with probability q, R selects a unilateral rule that does not select the voter's first choice alternative. If $p = \Omega(1)$ or $q = \Omega(1)$, then R obtains an approximation ratio of $\Omega(nd_{Ave}(m))$.*

Proof. The proof of Theorem 3 employs Yao's Minimax principle, to that end we define a randomized procedure for generating preference profiles. The procedure constructs a preference profile as follows:

1. Let $n = n' + 1$, $m \geq 4$, and let $m - 1$ divide n'. Select $w \in A$ uniformly at random.

2. Choose permutations π over $A \setminus \{w\}$ and σ over V uniformly at random.

3. Define the preferences of the voters as follows. For $i \in \{1, \cdots, n'\}$, $v_{\sigma(i)}$ ranks alternative $\pi(k)$ in position $\pi(k+i)+1$ (where the addition $k+i$ is modulo $m-1$) and ranks w first. Voter $v_{\sigma(n)}$ ranks w second and ranks the other alternatives arbitrarily.

Let E be a random election drawn from the above distribution. By construction, there is some $w \in A$ that is ranked first by all but one voter. Hence, the distance score of w is $r_d(2)$, as w is ranked second by one voter and the norm is l_1. For fixed distance function d, $r_d(2) = \Theta(1)$. Also, since $m - 1$ divides n', every other alternative is ranked in position k by at least $\frac{n'}{m-1}$ voters, for each $k = 2, \cdots, m$, and is ranked first by at most one voter. Hence, every alternative other than w achieves a distance score of at least $r_d(1) + \frac{n'}{m-1} \cdot \sum_{k=2}^{m} r_d(k) = \Theta(n \cdot d_{Ave}(m))$, since $r_d(1) = 0$.

Consider first the case where $p = \Omega(1)$. That is, when R selects a duple rule D with at least constant probability. Since w is selected from A at random, the probability that $w \in D$ is at most $\frac{2}{m}$. Hence, with probability at least $\frac{m-2}{m} \geq \frac{1}{3}$, an alternative with distance score $\Theta(n \cdot d_{Ave})$ is returned. Therefore the approximation ratio of R is at least $\Theta(pn \cdot d_{Ave}) = \Omega(nd_{Ave}(m))$.

Now consider the case where $q = \Omega(1)$. Since w is ranked first by all but one alternative and the voter that does not rank w first is selected uniformly at random, with probability $q \cdot \frac{n-1}{n}$, R selects an alternative other than w with a distance score of $\Theta(n \cdot d_{Ave})$. Thus, again, the approximation ratio of R is at least $\Theta(q \cdot \frac{n-1}{n} \cdot nd_{Ave}) = \Omega(nd_{Ave})$. □

Roughly speaking, Theorem 3 shows that for a strategyproof randomized voting rule R to obtain a nontrivial approximation ratio to $(\mathcal{U}, \hat{d})$, R must, with probability tending toward 1 for increasing numbers of voters and alternatives, select an alternative that is ranked first by some voter.

Theorem 4. *No strategyproof randomized voting rule approximates $(\mathcal{U}, \hat{d}_{disc})$ (plurality) to a ratio less than $2 - \frac{2}{n}$.*

Proof. Define the following procedure for constructing random preference profiles:

1. Select $w \in A$ uniformly at random. For ease of exposition, assume the members of $A \setminus \{w\}$ are the first $m - 1$ integers: $0, \cdots, m - 2$.

2. Choose permutations π over $A \setminus \{w\}$ and σ over V uniformly at random.

3. For $i \in \{1, \cdots, n - 1\}$, $v_{\sigma(i)}$ ranks w first and ranks alternative k in position $\pi(k+i)+1$ (where the addition is modulo $m-1$). $v_{\sigma(n)}$ ranks alternative k in position $\pi(k)$ and ranks w last.

In any elections drawn from the given distribution, the distance score of w is 1 (as all but one voter ranks w first). The distance score of any other alternative is at least $n-1$.

First, consider a duple rule D. Since w is selected uniformly at random, the probability that $w \in D$ is at most $\frac{2}{m}$. Thus, the probability that an alternative other than w is selected is at least $\frac{m-2}{m} \geq \frac{1}{3}$. Therefore, with probability at least $\frac{1}{3}$, D selects an alternative other than w with distance score at least $n-1$, resulting in an approximation ratio of at least $\frac{n-1}{3}$.

Now consider a unilateral rule, U. Since U is unilateral, there exists a single voter v_i that determines the winner of the election under U. The probability that v_i ranks w first is $\frac{n-1}{n}$. Since w is ranked first or is ranked last by every voter, any unilateral rule maximizes the probability of selecting w when it always returns either the first or last ranked alternative of v_i. Hence, U maximizes the probability of selecting w when it always returns v_i's first choice alternative. Therefore, any unilateral rule U selects w with probability at most $\frac{n-1}{n}$ and with probability at least $\frac{1}{n}$, U selects an alternative other than w. Hence, the approximation ratio of U is at least $\frac{n-1}{n} + \frac{1}{n} \cdot \frac{n-1}{1} = 2 - \frac{2}{n}$. Thus, no strategyproof randomized voting rule approximates $(\mathcal{U}, \hat{d}_{disc})$ to a factor less than $2 - \frac{2}{n}$. □

The remainder of the presented lower bound proofs use the following procedure (or a slight variation) for creating a distribution, $\mathcal{P}$, on preference profiles.

1. Let n be even, $m \geq 4$, and let $m-1$ divide $\frac{n}{2}$. Select $w \in A$ uniformly at random. For ease of exposition, assume the members of $A \setminus \{w\}$ are the first $m-1$ integers: $0, \cdots, m-2$.

2. Choose permutations π over $A \setminus \{w\}$ and σ over V, uniformly at random.

3. For $i \in \{1, \cdots, \frac{n}{2}\}$, let $v_{\sigma(i)}$ rank w first and rank alternative k in position $\pi(k+i)+1$ (where the addition $k+i$ is modulo $m-1$). For $i \in \{\frac{n}{2}+1, \cdots, n\}$, let $v_{\sigma(i)}$ rank w second and rank alternative k in position $\pi(k+i)+2$ if $\pi(k+i) \neq 1$ and in position 1 otherwise.

Based on the above procedure, it is observed that the selected alternative w is "close" (under most natural definitions of distance) to being a consensus winner, since half of the voters rank it first and the other half second. Since every other alternative is ranked cyclically by the voters and $m-1$ divides evenly into $\frac{n}{2}$, each alternative other than w is ranked in position k by the same number of voters. Let $\mathcal{P}'$ be the distribution over preference profiles identical to $\mathcal{P}$, except that for $i \in \{\frac{n}{2}+1, \cdots, n\}$, $v_{\sigma(i)}$ ranks w last. Lemma 1 will be employed in many of the lower bound proofs.

Lemma 1. *Let $\mathcal{K}$ be a consensus class and let d be a distance over elections. Let E be an election drawn from $\mathcal{P}$ or $\mathcal{P}'$. If for each alternative $a \in A \setminus \{w\}$, $d(a) \geq d_{min}$, then the approximation ratio achieved by any strategyproof randomized voting rule to $(\mathcal{K}, d)$ is at least $\frac{1}{2} \cdot \frac{d_{min}}{d(w)}$.*

Proof. Let R be any strategyproof randomized voting rule. Let p be the probability that R selects a duple rule and $1-p$ the probability that R selects a unilateral rule.

Consider first a duple rule, D. Since w is selected uniformly at random, the probability that $w \in D$ is at most $\frac{2}{m}$. Hence, the expected distance approximation of the alternative selected is at least

$$\frac{2}{m} + \frac{m-2}{m}\frac{d_{min}}{d(w)} \geq \frac{1}{2}\frac{d_{min}}{d(w)},$$

since $m \geq 4$.

Now consider a unilateral rule, U. Since w is ranked first by exactly half of the voters and second by the other half (last by the other half in the case of $\mathcal{P}'$) and those voters that rank w first are randomly distributed amongst all voters, the probability that U selects w is at most $\frac{1}{2}$. Hence, the probability that U selects an alternative other than w is at least $\frac{1}{2}$. The expected distance approximation is at least:

$$\frac{1}{2} \cdot \frac{d(w)}{d(w)} + \frac{1}{2} \cdot \frac{d_{min}}{d(w)} > \frac{1}{2} \cdot \frac{d_{min}}{d(w)}.$$

Therefore, the approximation ratio obtained by R is at least $\frac{1}{2} \cdot \frac{d_{min}}{d(w)}$. □

Lemma 1 allows one to lower bound the approximation ratio achievable for a number of distance rationalizations by strategyproof randomized voting rules. In particular, for scoring rules with $\alpha_1 \neq \alpha_2$, Theorem 5 obtains a lower bound close to the upper bound obtained by Corollary 1.

Theorem 5. *If R_α is a positional scoring rule with $\alpha_1 \neq \alpha_2$, then no strategyproof randomized voting rule approximates $(\mathcal{U}, \hat{d}_\alpha)$ to within a factor less than*

$$\frac{\alpha_1}{\alpha_1 - \alpha_2} - \frac{S - \alpha_2}{(m-1)(\alpha_1 - \alpha_2)},$$

where $S = \sum_{k=1}^{m} \alpha_k$.

Proof. Let E be an election drawn from $\mathcal{P}$. Note that the distance score of w is $n(\alpha_1 - \alpha_2)$. Let $k \in \mathbb{N}$ such that $k(m-1) = \frac{n}{2}$. Such a k exists, as $m-1$ divides $\frac{n}{2}$. Every alternative $a \in A \setminus \{w\}$ is ranked first k times, second k times, and in position $i \geq 3$, $2k$ times. If voter v_i ranks a in position r, then v_i contributes $2(\alpha_1 - \alpha_r)$ to the distance score of a, since the distance to the closest preference order to v_i in which a is ranked first is $2(\alpha_1 - \alpha_r)$. Thus, for every $a \in A \setminus \{w\}$

$$\begin{aligned}
d(a) &= k\left[2(\alpha_1 - \alpha_2)\right] + 2k\left[\sum_{i=3}^{m} 2(\alpha_1 - \alpha_i)\right] \\
&\geq 2k\left[2(m-2)\alpha_1 - 2\sum_{i=2}^{m}\alpha_i\right] \\
&= 2k\left[2(m-2)\alpha_1 - 2(S - \alpha_1)\right] \\
&= 4k(m-1)\alpha_1 - 4kS \\
&= 2n\alpha_1 - 2\left(\frac{n}{m-1}\right)S
\end{aligned}$$

By Lemma 1, the approximation ratio of any strategyproof randomized voting rule is lower bounded by

$$\frac{1}{2}\frac{2n\alpha_1 - 2\left(\frac{n}{m-1}\right)S}{n(\alpha_1 - \alpha_2)} = \frac{\alpha_1}{\alpha_1 - \alpha_2} - \frac{S - \alpha_2}{(m-1)(\alpha_1 - \alpha_2)}.$$

□

Theorem 6. *The approximation ratio obtained by any strategyproof randomized voting rule to $(\mathcal{U}, \hat{d}_{swap})$ (Borda) is $\Omega(m)$.*

Proof. Notice that in an election E drawn from $\mathcal{P}$, $d(w) = \frac{n}{2}$ since w is ranked first by $\frac{n}{2}$ voters and second by $\frac{n}{2}$ voters. Let $k(m-1) = \frac{n}{2}$. For any other alternative $a \in A$, a is ranked in position 1 and 2, k times and in position $r = 3, \cdots, m$, $2k$ times. Each voter that ranks a in position r contributes $r-1$ to $d(a)$, as a must be swapped with at least $r-1$ alternatives in order for a to be ranked first. Hence

$$\begin{aligned} d(a) &= k \cdot 1 + 2k \sum_{r=3}^{m} (r-1) \\ &\geq \frac{1}{2} \cdot \frac{n}{m-1} \sum_{r=2}^{m} (r-1) \\ &= \frac{1}{2} \cdot \frac{n}{m-1} \frac{(m-1)m}{2} = \Omega(nm). \end{aligned}$$

By Lemma 1 every strategyproof randomized voting rule achieves an approximation ratio of $\Omega(m)$ on $(\mathcal{U}, \hat{d}_{swap})$. □

Lemma 2. *Let E be an election drawn from $\mathcal{P}'$ and let $a \in A \setminus \{w\}$. There exists a set $A_a \subseteq A \setminus \{w\}$ with $|A_a| = \Theta(m)$ such that at least $\frac{3n}{4}$ voters prefer all members of A_a to a.*

Proof. Let E be an election drawn from $\mathcal{P}'$. Let π and σ be the random permutations over $A \setminus \{w\}$ and V used to construct E. Let $k \in \mathbb{N}$ such that $k(m-1) = \frac{n}{2}$. Since $m-1$ divides $\frac{n}{2}$, such a k exists. Among the voters, $v_{\sigma(1)}, \cdots, v_{\sigma(\frac{n}{2})}$, a is ranked last k times. Let $v_{\sigma(i_1)}, \cdots, v_{\sigma(i_k)}$ be the k voters that rank a last among the voters $v_{\sigma(1)}, \cdots, v_{\sigma(\frac{n}{2})}$. By construction, all the voters $v_{\sigma(i_j)}$ have the same preference order. Let A_a be the set of $\frac{m}{10}-1$ alternatives that immediately precede a in the voters $v_{\sigma(i_j)}$'s preference order. That is, A_a consists of those alternatives that are ranked among the bottom $\frac{m}{10}$ positions, other than a. For each i_j, by construction, voter $v_{\sigma(i_j-l)}$ ranks a in position $m-l$, for $l < m-1$. Thus, all alternatives in A_a are ranked above a by the voters $v_{\sigma(i_j)-l}$, for each $0 \leq l \leq m - \frac{m}{10} = \frac{9m}{10}$. Therefore among the voters $v_{\sigma(1)}, \cdots, v_{\sigma(\frac{n}{2})}$, there are $k \cdot \frac{9m}{10} \geq \frac{9(m-1)k}{10} = \frac{9n}{20}$ voters that rank all alternatives of A_a above a.

A similar argument applies to the voters among $v_{\sigma(\frac{n}{2}+1)}, \cdots, v_{\sigma(n)}$ that rank a second to last. Hence, among the voters $v_{\sigma(\frac{n}{2}+1)}, \cdots, v_{\sigma(n)}$, all members of A_a are ranked above a by at least $\frac{9n}{20}$ voters. Hence, the number of voters that prefer all members of A_a to a is greater than $\frac{3n}{4}$. □

Theorem 7. *The approximation ratio obtained by any strategyproof randomized voting rule to $(\mathcal{C}, \hat{d}_{disc})$ is $\Omega(n)$.*

Proof. Let E be an election drawn from $\mathcal{P}'$. By Lemma 2, for any alternative $a \in A \setminus \{w\}$ to become the Condorcet winner, $\Omega(n)$ voters must change their vote. However, for w to become the Condorcet winner, only one voter must change their vote. By Lemma 1, the approximation ratio of any strategyproof randomized voting rule is $\Omega(n)$. □

Theorem 7 roughly shows that no strategyproof randomized voting rule can outperform uniform random selection of an alternative, since under $\hat{d}_{disc}$ the maximum distance score of any alternative is at most n.

Theorem 8. *The approximation ratio obtained by any strategyproof randomized voting rule to $(\mathcal{C}, \hat{d}_{swap})$ (Dodgson) is $\Omega(n)$.*

Proof. Let E be an election drawn from $\mathcal{P}'$. To make a the Condorcet winner, a must be swapped with at least $|A_a| = \Theta(m)$ alternatives in the preferences of $\Omega(n)$ voters. To make w the Condorcet winner, it suffices that w be swapped with all the alternatives in the preference order of one voter that ranks w last. Thus, $d(w) = O(m)$. By Lemma 1, every strategyproof randomized voting rule obtains an approximation ratio of $\Omega(n)$. □

Theorem 9. *The approximation obtained by any strategyproof randomized voting rule to $(\mathcal{C}, d_{ins})$ (Maximin) is $\Omega(n)$.*

Proof. Recall that the d_{ins} score of an alternative a is equal to the minimum number of voters that must be added to make a the Condorcet winner. Let E be an election drawn from $\mathcal{P}'$. The d_{ins} score of w is 1, since the addition of a single voter that ranks w first will make w the Condorcet winner. However, the d_{ins} score of every other alternative is $\Omega(n)$, since to make any alternative $a \in A \setminus \{w\}$ the Condorcet winner $\Omega(n)$ voters must be added that rank a the members of A_a. Hence, by Lemma 1, the approximation ratio obtained by any strategyproof randomized voting rule is $\Omega(n)$. □

Notice that any alternative a can be made the Condorcet winner by the addition of at most $n+1$ voters that rank a first. Hence, $(\mathcal{C}, d_{ins})$ cannot be nontrivially approximated by any strategyproof randomized voting rule.

Theorem 10. *The approximation ratio obtained by any strategyproof randomized voting rule to $(\mathcal{M}, \hat{d}_{disc})$ is $\Omega(n)$.*

Proof. Let E be an election drawn from $\mathcal{P}$. To make w the majority winner, a single voter that ranks w second must change his vote. Every other alternative is ranked first by $\frac{n}{2(m-1)}$ voters. Hence, to make any other alternative the majority winner, at least $\frac{n}{2} + 1 - \frac{n}{2(m-1)} = \Omega(n)$ votes must be changed. By Lemma 1, every strategyproof randomized voting rule obtains an approximation ratio of $\Omega(n)$. □

Theorem 11. *Plurality is $(\mathcal{M}, \hat{d}_{disc})$-rationalizable.*

Proof. Clearly, a majority winner is also the plurality winner. Assume there is no majority winner. For alternative $a \in A$, the distance from E to the closest election in which a is a majority winner is $\lfloor \frac{n}{2} \rfloor + 1 - sc(a)$ (where $sc(a)$ is the plurality score of alternative a). Hence, the alternative with the highest plurality score also has the lowest distance. □

The distance rationalizations $(\mathcal{U}, \hat{d}_{disc})$ and $(\mathcal{M}, \hat{d}_{disc})$ have a large difference in approximation ratios achievable by strategyproof randomized voting rules even though they implement the same voting rule.

Procaccia [15] showed that veto can be approximated well with respect to maximizing the selected alternative's score. We show that veto cannot be approximated well with respect to minimizing the number of vetoes.

Let $\mathcal{V}$ be the consensus class consisting of elections in which some alternative is not vetoed. The consensus winner(s) are those alternatives that receive no vetoes.

Theorem 12. *Veto is $(\mathcal{V}, \hat{d}_{disc})$-rationalizable.*

Proof sketch. It suffices to observe that the distance score of an alternative a is simply the number of vetoes that it receives. If alternative a has fewer vetoes than alternative b, then the distance score of a is less than that of b. $\square$

For $a \in A$, let $v(a)$, be the number of voters that veto a. Since the total number of vetoes by all voters is n, uniform randomly selecting an alternative results in an approximation ratio of $\frac{1}{m}\sum_{a\in A} v(a) = \frac{n}{m}$.

Theorem 13. *The approximation ratio obtained by any strategyproof randomized voting rule to* $(\mathcal{V}, \hat{d}_{disc})$ *is* $\Omega(\frac{n}{m})$.

Proof sketch. Let $\mathcal{P}''$ be a distribution over preference profiles constructed similarly to $\mathcal{P}$, except a single voter, selected uniformly at random, ranks w last. Let E be an election drawn from $\mathcal{P}''$. Since w is vetoed by a single voter, $d(w) = 1$. However, each $a \in A \setminus \{w\}$ is vetoed by $\Omega(\frac{n}{m})$ voters. The Theorem then follows by Lemma 1. $\square$

4. CONCLUSIONS

This paper explores the idea of measuring the approximation ratio achieved by a strategyproof randomized voting rule with respect to a particular distance rationalization. Indeed, if a particular voting rule is employed in a given domain due to a domain specific distance rationalization, then the most natural measure of approximation is with respect to that rationalization.

This paper shows that the unanimity consensus class can be approximated well for a large class of distances by a single strategyproof randomized voting rule. It is shown that the Random Dictator voting rule (select the first choice alternative of a randomly selected voter) nontrivially approximates a large number of distance rationalizations with respect to unanimity. For a number of these distances, nearly tight lower bounds are presented. It is shown that deviating too greatly from the Random Dictator rule results in a trivial approximation ratio (i.e., the ratio obtained by ignoring the preference profile and selecting a random alternative).

The outlook for consensus classes, other than unanimity is bleaker. It is also shown that no strategyproof randomized voting rule nontrivially approximates many distance rationalizations with respect to the Majority and Condorcet consensus classes.

There exist a number of other distance rationalizations of common voting rules other than those considered in this paper. For example, the Copeland rule selects the alternative that maximizes the number of pairwise elections it wins and is rationalizable with respect to the Condorcet consensus class [13]. Future work will investigate the quality of strategyproof approximations obtainable to such rationalizations.

Another line of future work is to consider rationalization frameworks other than distance rationalization. For example, under the maximum likelihood estimation framework, one can measure the approximation ratio achieved by a randomized voting rule as a function of the likelihood of the alternative selected by that rule to being the true winner compared to the likelihood of the actual winner.

5. REFERENCES

[1] Eleanor Birrell and Rafael Pass. Approximately strategy-proof voting. In *Proceedings of the 22nd International Joint Conference on Artificial intelligence*, pages 67–72, 2011.

[2] Vincent Conitzer and Tuomas Sandholm. Universal voting protocol tweaks to make manipulation hard. In *Proceedings of the 18th International Joint Conference on Artificial intelligence*, pages 781–788, 2003.

[3] Vincent Conitzer and Tuomas Sandholm. Nonexistence of voting rules that are usually hard to manipulate. In *Proceedings of the 21st National Conference on Artificial intelligence*, pages 627–634. AAAI Press, 2006.

[4] Edith Elkind, Piotr Faliszewski, and Arkadii Slinko. On distance rationalizability of some voting rules. In *Proceedings of the 12th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 108–117. ACM, 2009.

[5] Edith Elkind, Piotr Faliszewski, and Arkadii Slinko. On the role of distances in defining voting rules. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 375–382, 2010.

[6] Edith Elkind, Piotr Faliszewski, and Arkadii Slinko. Homogeneity and monotonicity of distance-rationalizable voting rules. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, pages 821–828, 2011.

[7] Edith Elkind, Piotr Faliszewski, and Arkadii Slinko. Rationalizations of condorcet-consistent rules via distances of hamming type. *Social Choice and Welfare*, pages 1–15, 2011.

[8] Edith Elkind and Helger Lipmaa. Hybrid voting protocols and hardness of manipulation. In *In Proceedings of the 16th International Symposium on Algorithms and Computation*, pages 206–215. Springer-Verlag, 2005.

[9] Piotr Faliszewski and Ariel D. Procaccia. Ai's war on manipulation: Are we winning. *AI Magazine*, 31(4):53–64, 2010.

[10] Ehud Friedgut, Gil Kalai, and Noam Nisan. Elections can be manipulated often. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 243–249, 2008.

[11] Allan Gibbard. Manipulation of voting schemes. *Econometrica*, 41(4):587–601, 1973.

[12] Allan Gibbard. Manipulation of schemes that mix voting with chance. *Econometrica*, 25(3), 1977.

[13] Christian Klamler. The Copeland rule and condorcet's principle. *Economic Theory*, 25:745–749, 2005.

[14] Tommi Meskanen and Hannu Nurmi. Closeness counts in social choice. In *Power, Freedom, and Voting*. Springer-Verlag, 2008.

[15] Ariel Procaccia. Can approximation circumvent Gibbard-Satterthwaite? In *Proceedings of the 24th Conference on Artificial Intelligence*, pages 836–841, 2010.

[16] M. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10(2):187–217, 1975.

[17] Andrew Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 222–227, 1977.

Campaigns for Lazy Voters: Truncated Ballots

Dorothea Baumeister
Institut für Informatik
Universität Düsseldorf
Düsseldorf, Germany
baumeister@cs.uni-duesseldorf.de

Piotr Faliszewski
AGH University of
Science and Technology
Krakow, Poland
faliszew@agh.edu.pl

Jérôme Lang
LAMSADE
Université Paris-Dauphine
Paris, France
lang@lamsade.dauphine.fr

Jörg Rothe
Institut für Informatik
Universität Düsseldorf
Düsseldorf, Germany
rothe@cs.uni-duesseldorf.de

ABSTRACT

We study elections in which voters may submit partial ballots consisting of truncated lists: each voter ranks some of her top candidates (and possibly some of her bottom candidates) and is indifferent among the remaining ones. Holding elections with such votes requires adapting classical voting rules (which expect complete rankings as input) and these adaptations create various opportunities for candidates who want to increase their chances of winning. We provide complexity results regarding planning various kinds of campaigns in such settings, and we study the complexity of the possible winner problem for the case of truncated votes.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Theory

Keywords

elections, manipulation, possible winner, bribery

1. INTRODUCTION

Elections and voting constitute an important mechanism for aggregating preferences of independent agents (be it nations choosing their leaders, people recommending movies, or software agents planning their joint actions). In the standard model of voting, we are given some set of candidates C and each agent (that is, each voter) ranks all the candidates in C from the most preferred one to the most despised one. Then, a voting rule is used to find the winner(s). Unfortunately, ranking all candidates is feasible only if there are very few candidates, and even then the voters might be unwilling to provide full rankings. Indeed, most political elections are held using the plurality rule, which asks each voter to name the favorite candidate only, and elects whoever gets the most votes. Also, elections over large combinatorial domains (such as those encountered, for example, in multiagent planning settings) require the use of nontrivial representation languages to express preference orders.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

One of the most natural solutions to the problem of overburdening the voters with ranking too many candidates is to allow them to cast truncated preference orders. Indeed, each voter is likely to know who are her most favorite candidates and if she is unwilling to put the effort into ranking the remaining ones, it is safe to assume that she likes them less than the ranked ones but is otherwise indifferent among them. We will call such preferences "top-truncated."

On the other hand, it is possible that a voter is indifferent among a large set of acceptable candidates, but truly hates some remaining ones (compare this with the idea of destructive manipulation and control [10, 23]). Then, we would say that this voter has "bottom-truncated" preferences. Finally, it is also possible that a voter would have strong preferences regarding a small number of her top candidates and regarding a small number of her bottom candidates, but would be indifferent regarding the large group of "middle-ranking" candidates. We refer to such votes as "doubly-truncated."

Although allowing truncated votes does not solve all problems with ranking large candidate sets (for example, it seems completely inappropriate for voting in large combinatorial domains), it certainly is a very good solution for *some* settings, both in political elections (for example, top-truncated ballots are allowed in political elections in Slovenia) and for software agents (for example, if one builds a meta-search engine using voting techniques [12], where the votes—search engine results for a given query—are necessarily top-truncated).

Unfortunately, typical voting rules, such as, e.g., Borda (defined formally in Section 2) inherently depend on voters providing complete rankings and have to be adapted for the case of truncated votes. For example, for Borda, we could assume that each unranked candidate receives 0 points from a given vote (a method used in Slovenia, which we will call the *pessimistic* scoring model). Or, if there are m candidates but a vote ranks only k of them, then the ranked candidates get $m-1,\dots,m-k$ points (depending on their position in the ranking) and each unranked candidate gets $m-k-1$ points. This method is sometimes called *modified Borda* (see, e.g., [15]) and is used, e.g., by the Irish Green Party to choose its leader. We will call this method the *optimistic* scoring model; optimistic scoring has the advantage that it provides an incentive for voters to rank more candidates.

Given the two above variants of Borda, it is immediately clear that a candidate can benefit from convincing some of the agents to extend their votes. Under pessimistic scoring a candidate should try to get as many voters as possible to add him or her to the ranking; under optimistic scoring the situation is more complicated (see Theorems 3.4 and 3.5).

Campaigns aimed at extending truncated ballots are particularly attractive because they can be presented as "enhancing voters' awareness" and as "providing voters with an incentive to cast their votes." Also, they are less "invasive" than manipulation or bribery actions (see, e.g., [19, 21, 18]), as they do not aim at *changing* voters' preferences but only at *extending* them. Thus, such campaigns are viewed as inherently positive. We study the computational complexity of such campaigns in Section 3.

The fact that standard voting rules have to be modified for the case of truncated votes can sometimes be viewed as too demanding. In such situations election rules force the voters to provide complete rankings. However, even then the reasons why truncated preferences arise still apply. As a result, it is likely that voters still have truncated preferences and simply complete them arbitrarily at the time of voting. From the point of view of candidates, it is interesting to know, given truncated votes, for which candidates are there completions of the votes that ensure their victory. We study such scenarios in Section 4, where we consider the possible winner problem for the case of truncated votes.

We conclude the paper by describing related work and by presenting future research directions in Sections 5 and 6.

2. PRELIMINARIES

Elections with truncated ballots. An election is a pair $E = (C,V)$, where $C = \{c_1,\dots,c_m\}$ is a set of candidates and $V = (v_1,\dots,v_n)$ is a collection of voters. Each voter is represented via her preferences over the set C. There are many ways in which a voter's preferences can be modeled. Throughout this paper we use a variant of the ordinal model, where each voter's preferences are represented via a (possibly partial) order over the set of candidates. We will refer to this order either as a *preference order*, a *ballot*, or, slightly abusing notation, a *vote*; we use these terms essentially interchangeably. For example, if $C = \{c_1,c_2,c_3\}$, a voter who prefers c_1 to c_2 and c_2 to c_3 (and, thus, has *complete* preferences) would have preference order $c_1 \succ c_2 \succ c_3$. We also allow the voters to have partial preference orders. In particular, we focus on the following three classes of such votes. Let t and b be two nonnegative integers such that $t+b \leq \|C\|$:

Doubly-truncated votes. A partial preference order $\succ$ on C is (t,b)-*doubly-truncated* if there is a permutation π over $\{1,\dots,\|C\|\}$ such that $\succ$ is of the form $c_{\pi(1)} \succ \cdots \succ c_{\pi(t)} \succ \{c_{\pi(t+1)},\dots,c_{\pi(m-b)}\} \succ c_{\pi(m-b+1)} \succ \cdots \succ c_{\pi(m)}$ (i.e., each candidate in the set $\{c_{\pi(t+1)},\dots,c_{\pi(m-b)}\}$ is strictly below $c_{\pi(t)}$, strictly above $c_{\pi(m-b+1)}$, but the voter is indifferent among the members of the set; we refer to candidates $c_{\pi(1)},\dots,c_{\pi(t)},c_{\pi(m-b+1)},\dots,c_{\pi(m)}$ as the *ranked* candidates, and to the remaining ones as *unranked*). For a (t,b)-doubly-truncated preference order $\succ$ we define $\mathrm{top}(\succ) = t$ and $\mathrm{bottom}(\succ) = b$.

Top-truncated votes. A partial preference order is *t-top-truncated* if it is $(t,0)$-doubly-truncated.

Bottom-truncated votes. A partial preference order is *b-bottom-truncated* if it is $(0,b)$-doubly-truncated.

We say that a preference order is *doubly-truncated* (*top-truncated*, *bottom-truncated*) if there are values t and b for which it is (t,b)-doubly-truncated (t-top-truncated, b-bottom-truncated). Similarly, we say that an election $E = (C,V)$ is *doubly-truncated* (*top-truncated*, *bottom-truncated*) if each vote in V is doubly-truncated (top-truncated, bottom-truncated). We say that an election is (at-most-t)-top-truncated if for each vote v in E, there is an integer $t_v \leq v$ such that v is t_v-top-truncated.

We use the following, somewhat subtle, notation to describe truncated votes. Let C be a set of candidates. If in a preference order we write $\overleftarrow{S}$, where S is a subset of C, then we mean listing all members of S in some fixed (easily computable) order. If we write $\overrightarrow{S}$, then we mean listing all members of S in the reverse of this order. If we write S (without any arrows on top), we mean that S are the unranked candidates. For example, if T and B are two disjoint subsets of C, then by $\overleftarrow{T} \succ C \setminus (T \cup B) \succ \overleftarrow{B}$ we mean a $(\|T\|,\|B\|)$-doubly-truncated preference order where candidates from T are ranked at the top of the vote (in some order), candidates in B are ranked at the bottom of the vote (in some order), and the remaining candidates in the middle are unranked.

Voting Rules for Top-Truncated Votes. A voting rule $\mathscr{R}$ maps an election $E = (C,V)$ to a set $\mathscr{R}(E) \subseteq C$ of candidates. We allow a voting rule to output more than one winner or no winner at all. Unfortunately, standard definitions of many voting rules assume that the voters have complete preferences and it may not be completely obvious how to adapt them to the case of truncated rules (see, e.g., how Brams and Sanver [7] obtained fallback voting, which accepts top-truncated orders, from Bucklin voting, which requires complete orders). We now describe how we adapt several well-known (families of) voting rules to top-truncated ballots.

Let $E = (C,V)$ be an election with candidate set $C = \{c_1,\dots,c_m\}$ and voter collection $V = (v_1,\dots,v_n)$, where each vote is top-truncated. A scoring vector $\alpha = (\alpha_1,\dots,\alpha_m)$ is a vector of nonnegative integers such that $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_m$. If all votes are complete, then under scoring rule $\mathscr{R}_\alpha$, each candidate $c_j \in C$ receives α_i points for each vote where c_j is ranked on the i'th position. The winners of the election are the candidates with most points. Typically, we consider families of scoring rules, with one scoring vector for each possible number of candidates.

For example, for each positive integer k, *k-approval* uses vectors of the form $(1,\dots,1,0,\dots,0)$ with k ones; *plurality voting* is 1-approval; and *Borda* uses vectors of the form $(m-1,m-2,\dots,0)$, where m is the number of candidates.

If (some of) the votes are top-truncated then we modify this point-assignment procedure as follows. For a t-top-truncated vote $c_{\pi(1)} \succ \cdots \succ c_{\pi(t)} \succ \{c_{\pi(t+1)},\dots,c_{\pi(m)}\}$, each ranked candidate $c_{\pi(i)}$, $1 \leq i \leq t$, receives α_i points, and each unranked candidate receives s points, where $s = \alpha_m$ in the pessimistic scoring model and $s = \alpha_{t+1}$ in the optimistic scoring model. Note that the optimistic scoring model, when applied to Borda, is equivalent to what is known as *modified Borda*; see, e.g., [15].

The pessimistic model is the most popular one in practice. It is used, for example, in Slovenia and in Kiribati for truncated ballots under Borda's rule. One of the downsides of the pessimistic model is that it gives incentives for voters to rank only a single candidate (so the impact of the vote on the score of this candidate, relative to the scores of other candidates, is greatest). On the other hand, the optimistic model rewards the voters who rank more candidates: the more candidates one ranks, the more points (in relative terms) these candidates receive.

Compared to the case of scoring rules, voting rules based on head-to-head comparisons of candidates are much easier to adapt. Let $E = (C,V)$ be an election with candidate set $C = \{c_1,\dots,c_m\}$ and voter collection $V = (v_1,\dots,v_n)$, where the voters may have truncated votes. For each two candidates c_i, c_j, we define $N_E(c_i,c_j) = \|\{k \mid v_k \text{ prefers } c_i \text{ to } c_j\}\|$. Note that if all votes are complete, then for each distinct $c_i,c_j \in C$ it holds that $N_E(c_i,c_j) + N_E(c_j,c_i) = n$. However, when some votes are truncated then for some $c_i,c_j \in C$ it is the case that $N_E(c_i,c_j) + N_E(c_j,c_i) < n$ (because some voters are, effectively, indifferent between c_i and c_j).

Under the *Copeland rule*, the score of a candidate $c_i \in C$ is defined as $\|\{c_j \in C \setminus \{c_i\} \mid N_E(c_i,c_j) > N_E(c_j,c_i)\}\| + (1/2)\,\|\{c_j \in C \setminus \{c_i\} \mid N_E(c_j,c_i) = N_E(c_i,c_j)\}\|$. However, we will focus on a related rule, called *Copeland*^{0}, where the score of a candidate $c_i \in C$ is defined to be $\|\{c_j \in C \setminus \{c_i\} \mid N_E(c_i,c_j) > N_E(c_j,c_i)\}\|$ (i.e., defeating a candidate in a head-to-head contest gives one point, but losing and tieing both have no effect on the score). Under *maximin*, the score of a candidate $c_i \in C$ is $\min_{c_j \in C \setminus \{c_i\}} N_E(c_i,c_j)$. For both rules, the winners are the candidates with the highest score. Given an election $E = (C,V)$, a candidate $c \in C$, and a voting rule $\mathscr{R}$ that assigns scores to candidates, we write $\mathrm{score}(c)$ to denote the score of candidate c. The election and the voting rule will always be clear from context.

Computational Complexity and Algorithms. We assume familiarity with standard notions of complexity theory such as the classes P and NP, and the notions of many-one polynomial-time reducibility, NP-completeness, and NP-hardness. We also assume familiarity with parameterized complexity theory and classes FPT and W[1]. We point the reader to the textbook [28] for references.

We will need the following NP-complete problems.

PARTIAL-SET-MULTICOVER	
Given:	A base set $B = \{b_1,\dots,b_m\}$, each element b_i of B paired with a positive integer $\mathrm{req}(b_i)$ (the covering requirement of b_i), a family $\mathscr{S} = \{S_1,\dots,S_n\}$ of subsets of B, each set S_j paired with a positive integer $\mathrm{cost}(S_j)$, a nonnegative integer K (the budget), and a nonnegative integer k (the covering request).
Question:	Is there a set $A \subseteq \{1,\dots,n\}$ and a set $C \subseteq B$ such that: (a) for each $b_i \in C$, $\|\{j \in A \mid b_i \in S_j\}\| \geq \mathrm{req}(b_i)$ (that is, for each element of C the sets S_j, $j \in A$, jointly satisfy its covering requirement), (b) $\|C\| \geq k$ (that is, C satisfies the covering request), and (c) $\sum_{j \in A} \mathrm{cost}(S_j) \leq K$ (that is, we do not exceed the budget)?

PARTIAL-SET-MULTICOVER is the most general of a family of related problems. In PARTIAL-SET-COVER, each covering requirement is set to 1. In SET-MULTICOVER, we have to cover all elements in the base set (i.e., $k = \|B\|$). In SET-COVER, each covering requirement is set to 1 and $k = \|B\|$. Finally, in X3C we have the same setting as for SET-COVER, but $\|B\|$ is a multiple of 3, each set in $\mathscr{S}$ has exactly 3 elements and unit cost, and $K = \|B\|/3$. Except for X3C, each of these problems has a natural minimization variant where we seek to minimize the total cost of the selected sets.

We will also use the following standard minimization problem.

KNAPSACK	
Input:	Nonnegative integers $w_1,\dots,w_n$ (the weights) and $v_1,\dots,v_n$ (the values) and a nonnegative integer T (the target value).
Output:	A set $A \subseteq \{1,\dots,n\}$ such that (a) $\sum_{i \in A} v_i \geq T$ and (b) $\sum_{i \in A} w_i$ is minimal (or indication that such a set does not exist).

A *minimization problem A* is a problem that asks us to compute some solution s that minimizes a certain cost function $\mathrm{cost}(s)$. For a given instance I, we write $\mathrm{OPT}(I)$ to denote the value of a solution with minimal cost. For a given number $\alpha \geq 1$, we say that an algorithm $\mathscr{A}$ is an *α-approximation algorithm* for a given minimization problem if for each input instance I, $\mathscr{A}$ outputs a valid solution s such that $\mathrm{cost}(s) \leq \alpha \cdot \mathrm{OPT}(I)$. A *fully polynomial-time approximation scheme* (*FPTAS*) for a minimization problem is an algorithm that, given an instance I and a positive rational value ε, runs in time polynomial in $|I|$ and $1/\varepsilon$ (i.e., in time polynomial with respect to the length of the encoding of I and the value of $1/\varepsilon$), and outputs a solution s such that $\mathrm{cost}(s) \leq (1+\varepsilon)\mathrm{OPT}(I)$. It is well-known that there is an FPTAS for KNAPSACK and that the decision variant of KNAPSACK is NP-complete.

We will also study the possible (co-)winner problem, introduced by Konczak and Lang [25]. For a given voting rule $\mathscr{R}$, define:

$\mathscr{R}$-POSSIBLE-WINNER ($\mathscr{R}$-PW)	
Given:	An election $E = (C,V)$, where the ballots in V are partial orders over the set of candidates, and a distinguished candidate $p \in C$.
Question:	Is it possible to complete the votes in E so that p is an $\mathscr{R}$ winner?

An important special case of the possible winner problem is the unweighted coalitional manipulation problem, $\mathscr{R}$-UCM, where some voters (so-called *honest voters*) have complete preference orders, and some voters (so-called *manipulators*) have empty preference orders. Intuitively, in $\mathscr{R}$-UCM the manipulators are seeking possibly dishonest votes that would ensure their favorite candidate's victory; see [19, 21] for more details on $\mathscr{R}$-UCM.

3. CAMPAIGNING PROBLEMS

Let us now focus on campaign management problems that arise in the context of top-truncated votes. As we have noticed, a candidate may benefit from convincing some voters to extend their top-truncated preference orders. Naturally, some voters might be harder to affect than others. Thus, we should assume that each voter v has some function δ that describes the cost of extending v's top-truncated vote. However, since the voter is originally indifferent among the unranked candidates, it is reasonable to assume that the cost of extending the vote depends only on the number of added candidates and not on their names.

The task of the campaign manager is to figure out how (and to what extent) to extend the votes in order to ensure her candidate's victory, while spending as little as possible. Following the naming convention from two papers on campaign management in elections [13, 31], we call our problem EXTENSION-BRIBERY.

3.1 Formal Definition

We now give a more formal description of extension bribery. Let $E = (C,V)$ be a top-truncated election where $C = \{p, c_1, \dots, c_{m-1}\}$ and $V = (v_1,\dots,v_n)$. Let $\Delta = (\delta_1,\dots,\delta_n)$ be a collection of functions from nonnegative integers to nonnegative integers, such that for each i, $1 \leq i \leq n$, it holds that $\delta_i(0) = 0$ and δ_i is nondecreasing. We will refer to the functions in Δ as *extension bribery cost functions*. Let $E' = (C,V')$, $V' = (v'_1,\dots,v'_n)$ be a top-truncated election obtained from E by extending the votes in V. We define the cost of extending E to E' to be $\sum_{i=1}^{n} \delta_i(\mathrm{top}(v'_i) - \mathrm{top}(v_i))$. The goal is to find a minimal-cost extension of E that ensures p's victory. Note that in each vote that we extend by some k candidates, we are free to rank these k candidates in any way, provided that they all follow the originally ranked candidates.

For a given voting rule $\mathscr{R}$, define:

$\mathscr{R}$-EXTENSION-BRIBERY	
Given:	An election $E = (C,V)$, where the ballots in V are possibly top-truncated, a collection Δ of extension-bribery cost functions (one per voter), a distinguished candidate $p \in C$, and a nonnegative integer B (the budget).
Question:	Is there an extension of cost at most B of election E where p is an $\mathscr{R}$ winner?

Each cost function δ in Δ is represented by providing at most $\|C\|$ integer values, $\delta(0), \delta(1), \dots, \delta(\|C\|)$. Unless specified otherwise, all integers are encoded in binary. In particular, we study two special cases of extension bribery cost functions δ (inspired by cost functions from [14]). In the *zero-cost* model we take each cost function δ to be such that $\delta(k) = 0$ for each nonnegative integer k. In the *unit-cost* model, we take each function δ to be such that $\delta(k) = k$ for each nonnegative integer k (that is, there is a unit cost for extending each vote with a single candidate).

In the minimization variant of the problem the budget is not part of the input, and we simply ask if it is possible to ensure p's victory, and if so, what is the lowest cost at which this can be achieved.

Different families of cost functions correspond to different campaign settings. If we allow general cost functions, our problem models a campaign management scenario where affecting each voter may require a different amount of effort. The unit-cost model corresponds to settings where we have no knowledge of the difficulty of affecting particular voters and we simply minimize the number of additionally ranked candidates. The zero-cost model corresponds to settings where we want to find out if extension-bribery type campaign can succeed at all.

3.2 Checking the Possibility of Success

We consider the zero-cost model first. It turns out that in this case EXTENSION-BRIBERY is easy for maximin and scoring rules under the pessimistic scoring model.

THEOREM 3.1. EXTENSION-BRIBERY *under the zero-cost model is in* P *for maximin, and—under the pessimistic scoring model—for each efficiently computable family of scoring rules.*

PROOF. Let $\mathscr{R}$ be one of the voting rules from the theorem statement. Set $E = (C,V)$ to be our input top-truncated election and let p be the candidate whose victory we want to ensure. Irrespective of our choice of $\mathscr{R}$, the best we can do is to extend each top-truncated vote that does not yet rank p to include p. $\square$

On the other hand, under scoring rules and the optimistic scoring model already this very simplified variant of EXTENSION-BRIBERY can be NP-complete. The reason for this is that under optimistic scoring we have very strong side effects—adding a candidate to a vote decreases the score of the remaining unranked candidates. This means that under optimistic scoring with zero-costs the best action a campaign manager can take is to fully extend all votes. This, effectively, reduces the problem to the UCM problem.

THEOREM 3.2. *For each voting rule $\mathscr{R}$ that can be represented as a family of scoring rules, it holds that $\mathscr{R}$-*UCM *reduces to $\mathscr{R}$-*EXTENSION-BRIBERY *under optimistic scoring with zero-costs.*

PROOF. Let $I = (C,V,W,p)$ be an instance of $\mathscr{R}$-UCM, where C is a set of candidates, V is a collection of honest voters, W is a collection of manipulators, and $p \in C$ is our preferred candidate. We construct an instance $I' = (C,V',\Delta,p,0)$ of $\mathscr{R}$-EXTENSION-BRIBERY as follows: We set V' to be the concatenation of the lists V and W', where each manipulator in W is replaced in W' by a voter with 1-top-truncated vote $p \succ C \setminus \{p\}$, and we let Δ be a collection of zero-cost functions. The reader can verify that there is a solution for I if and only if there is a solution of zero cost for I'. $\square$

Since it is now known that Borda-UCM is NP-complete [11, 5], we immediately have that Borda-EXTENSION-BRIBERY under optimistic scoring is NP-complete as well, even in the zero-cost model. For Copeland^0 we also obtain NP-completeness in the zero-cost model via a reduction from Copeland^0-UCM (which is NP-complete [20]), but this time the reduction is more involved.

THEOREM 3.3. *Copeland*^{0}-EXTENSION-BRIBERY *is* NP-*complete, even in the zero-cost model.*

However, in a way, Theorems 3.2 and 3.3 are not satisfying; in either case our proofs use the fact that (almost) all voters rank (almost) all candidates. In realistic settings we would rather expect that almost all voters would have very short top-truncated votes. Thus, it is interesting to ask what happens for, say, Borda and Copeland^0 if our input election is restricted to contain (at-most-k)-top-truncated votes, for some small value of k. Answering this question seems nontrivial under the zero-cost model (with optimistic scoring, for Borda). In particular, for the case of Borda, this problem appears to be related to manipulation by more than two manipulators (even though there is a proof of Borda-UCM NP-completeness for the case of two manipulators, generalizing it to the case of more manipulators is not trivial [11, 5]). However, we can answer it for the case of the unit-cost model (see Theorem 3.4).

3.3 Minimizing the Campaign's Cost

After the campaign manager verifies that indeed it is possible to run a successful campaign, the next step is to find a campaign strategy that requires smallest effort. In particular, if the manager has little knowledge about the difficulty of affecting particular voters, her most reasonable approach is to simply minimize the degree to which she extends the votes. Formally, this is captured by the unit-cost model. Unfortunately, it turns out that even in this very simple model we reach broad hardness results.

THEOREM 3.4. EXTENSION-BRIBERY *under the unit-cost model is* NP-*complete for Borda (with optimistic scoring), maximin, and Copeland*^{0}*, even if each vote is (at-most-*6*)-top-truncated.*

PROOF. Let us consider the case of Borda first. We give a reduction from X3C. Let $I = (B,\mathscr{S})$ be our input instance where $B = \{b_1,\dots,b_{3k}\}$ and $\mathscr{S} = (S_1,\dots,S_n)$. Without loss of generality, we assume that k is odd.

We build an instance $I' = (C,V,\Delta,p,k)$ of Borda-EXTENSION-BRIBERY (note that in our instance the budget it set to k). We set $C = B \cup \{p,x,y\}$ and construct the voter collection as follows. First, for each S_i, $1 \le i \le n$, we introduce a voter v_i with vote $p \succ \overleftarrow{S_i} \succ C \setminus S_i$. Then, we add enough (but at most polynomially many) 6-top-truncated votes that ensure that for each j, $1 \le j \le 3k$, $\mathrm{score}(b_j) = \mathrm{score}(p) + k - 1$, and $\mathrm{score}(x) \le \mathrm{score}(p)$ and $\mathrm{score}(y) \le \mathrm{score}(p)$. We do so by using the following construction.

Fix some candidate $c \in C$. We define a collection $V(c)$ of voters to contain the following $(3k+3)/6$ pairs of voters. (To define this collection of voters, we rename the candidates so that $C = \{c,y,c_3,\dots,c_{3k+3}\}$.) The first pair contains 6-top-truncated votes $c \succ y \succ c_3 \succ c_4 \succ c_5 \succ c_6 \succ C \setminus \{c,y,c_3,c_4,c_5,c_6\}$ and $c_6 \succ c_5 \succ c_4 \succ c_3 \succ c \succ y \succ C \setminus \{c,y,c_3,c_4,c_5,c_6\}$. The following pairs of votes are constructed as follows. For each i, $1 \le i \le ((3k+3)/2) - 1$, let $C_i = \{c_{6i+1},c_{6i+2},c_{6i+3},c_{6i+4},c_{6i+5},c_{6i+6}\}$ and add a pair of 6-top-truncated votes $\overleftarrow{C_i} \succ C \setminus C_i$ and $\overrightarrow{C_i} \succ C \setminus C_i$. It is easy to see that within $V(c)$, it holds that for each candidate c_i, $3 \le i \le 3k+3$, we have $\mathrm{score}(c_i) = \mathrm{score}(c) - 1$, and that $\mathrm{score}(y) = \mathrm{score}(c) - 2$. (This is so because each candidate appears in the preference orders of the voters in exactly one pair; in the first pair c gets one point more than each of $c_3,\dots,c_6$, and y gets one point less than each of $c_3,\dots,c_6$. In the further pairs each candidate c_i gets as many points as each of the candidates $c_3,\dots,c_6$ in the first pair.) Thus, by grouping together sufficiently many (but not more than polynomially many) collections of voters of the form $V(c)$, for $c \in B \cup \{p,x\}$, we can satisfy the score requirements from the paragraph above.

We complete the construction of I' by setting Δ to be a collection of unit-cost functions.

We claim that if I is a *yes*-instance of X3C then there is an extension bribery of costs at most k that ensures p's victory. Let

$A \subseteq \{1,\ldots,3k\}$ be a solution for I, that is, a set such that (a) $\|A\| \leq k$ and (b) $\bigcup_{i \in A} S_i = B$. If we extend the votes v_i with $i \in A$ so that each of them also ranks x, then it will cost $\|A\|$, the scores of x and p will not change, and the score of each $b_i \in B$ will decrease by $k-1$ (because A corresponds to an exact cover of B). As a result, p will be a winner of the election.

For the other direction, we claim that if there is a solution to I' of cost at most k then I is a *yes*-instance of X3C. To see this, note that, by construction of (C,V), a successful extension bribery has to ensure that each $b_i \in B$ loses at least $k-1$ points. That is, we have to ensure that at least $3k(k-1)$ points are lost by the candidates in B. On the other hand, we can extend the votes by adding at most k candidates. Thus, on the average, each act of extending a vote has to, effectively, decrease the scores of $3k(k-1)/k = 3k-3$ candidates. However, by an easy argument, this is possible only if we extend some k votes of voters $v_1,\ldots,v_n$, each by adding either candidate x or candidate y (but not both). Further, to decrease the score of each candidate $b_i \in B$ by $k-1$, these k voters have to correspond to an exact cover of B.

The proofs for Copeland^0 and maximin proceed similarly. $\square$

On the other hand, for the case of scoring rules under the pessimistic scoring model, we can, for all practical purposes, handle essentially all cost models. As in the case of unit cost, the reason for this is that under pessimistic scoring the best one can do is to simply rank p in the votes that do not rank p yet. The only, fairly easily solvable, difficulty is that under general cost functions one has to decide which votes to extend.

THEOREM 3.5. *There is an algorithm that given a scoring vector $\alpha = (\alpha_1,\ldots,\alpha_m)$, an instance I of $\mathscr{R}_\alpha$-*EXTENSION-BRIBERY *in the pessimistic scoring model, and positive rational value ε, outputs (in time polynomial in the encoding size of I and $1/\varepsilon$) a solution s for I such that* $\mathrm{cost}(s) \leq (1+\varepsilon)\mathrm{OPT}(I)$.

PROOF. Let $I = (C,V,\Delta,p)$, where $C = \{p,c_1,\ldots,c_{m-1}\}$, $V = (v_1,\ldots,v_n)$, and $\Delta = (\delta_1,\ldots,\delta_n)$. Our algorithm works as follows. First, if p is already a winner then we output an empty solution and terminate. Otherwise, let c_j be one of the current winners. We have $\mathrm{score}(c_j) > \mathrm{score}(p)$. It is easy to see that it suffices to find a set $A \subseteq \{1,\ldots,n\}$ such that (a) for each $i \in A$, v_i does not rank p, (b) $\sum_{i \in A} \alpha_{\mathrm{top}(v_i)+1} \geq \mathrm{score}(c_j) - \mathrm{score}(p)$, and (c) $\sum_{i \in A} \delta_i(1)$ is minimal among all subsets satisfying the previous two conditions. Clearly, finding such a set A reduces to solving a knapsack problem (where the values $\delta_i(1)$ take the role of the weights and the values $\alpha_{\mathrm{top}(v_i)+1}$ take the role of the values). A standard FPTAS for knapsack proceeds either by scaling the weights or by scaling the values; in our case we use a version that scales the weights. $\square$

The above proof implies that R_α-EXTENSION-BRIBERY is in P when the cost functions are encoded in unary. For binary encoding we have the following result.

THEOREM 3.6. *If the scoring vector α is part of the input then $\mathscr{R}_\alpha$-*EXTENSION-BRIBERY*, under either of our two scoring models, is* NP*-complete. However, for each fixed scoring vector α, $\mathscr{R}_\alpha$-*EXTENSION-BRIBERY *is in* P.

The NP-completeness proof for the pessimistic scoring model follows by a simple reduction from KNAPSACK. P-membership follows by using the same techniques as in Theorem 4.15 in [18].

3.4 Dealing with the Hard Cases?

The above results show that in many practically important cases effective campaign management requires solving NP-complete problems. Are there ways in which we can deal with this problem? Depending on the voting rule and the particular setting we have several options. In this section we will focus on cases where the only legal vote extensions are those that add p to those votes that do not rank p yet. This is the most natural type of campaign to run and from a practical perspective it is most important to be able to solve EXTENSION-BRIBERY instances for this case (in addition, this strategy is optimal for maximin). Further, it seems that more general variants of EXTENSION-BRIBERY might require much more involved approaches than presented here.

The main source of hardness of EXTENSION-BRIBERY problems is their close relation to set-covering problems. We consider Copeland^0 first. Let $I = (C,V,\Delta,p,B)$ be an instance of Copeland^0-EXTENSION-BRIBERY. Let us assume that there is some candidate $c \in C, c \neq p$, who has the highest score among all candidates and whose score cannot be decreased by adding p to the truncated votes (at cost B). This means that, in order to win, p has to obtain $\mathrm{score}(c) - \mathrm{score}(p)$ additional points. That is, the task of the campaign manager is to find a subcollection V' of voters that do not rank p, such that (a) the total cost of adding p to each vote in V' is at most B, and (b) there is a group of at least $\mathrm{score}(c) - \mathrm{score}(p)$ candidates against whom p was losing-or-tieing the head-to-head contests prior to vote extension and against whom p is winning these head-to-head contests after the extension.

However, this simply means that the campaign manager has to solve PARTIAL-SET-MULTICOVER for an instance with the base set B equal to the set of candidates against whom p is losing-or-tieing the head-to-head contests, with the family of sets $\mathscr{S} = \{S(v) \mid \text{voter } v \text{ does not rank } p\}$ (where $S(v)$ is the subset of candidates in B that voter v does not rank; the cost of $S(v)$ is the cost of including p in v's preference order), with the covering requirement of each $d \in B$ being exactly the number of voters that would have to additionally rank p ahead of d for p to win the head-to-head contest, with the covering request k equal to $\mathrm{score}(c) - \mathrm{score}(p)$, and with the same budget as in the EXTENSION-BRIBERY instance.

Based on this observation, and with a simple construction (omitted due to space restriction), we see that if there were any way to efficiently solve Copeland^0-EXTENSION-BRIBERY then there also would be an analogous way to solve PARTIAL-SET-MULTICOVER. Unfortunately, it seems that PARTIAL-SET-MULTICOVER is a particularly difficult NP-complete problem: No nontrivial approximation algorithm for it is known (even though SET-COVER and SET-MULTICOVER [22] have natural $O(\log m)$-approximation algorithms, where m is the size of the base set). One might hope that the problem would at least be fixed-parameter tractable for the parameter "number of base-set elements to cover" (corresponding to cases where p has a score close to that of the current winner) as for this case PARTIAL-SET-COVER is in FPT [6]. Unfortunately, we have the following result, which implies Corollary 3.8.

THEOREM 3.7. *The problem* PARTIAL-SET-MULTICOVER *is* W[1]*-complete for the parameter (k,B), where k is the number of base-set elements to cover and B is the budget.*

PROOF SKETCH. The proof of hardness proceeds via a reduction from the problem CLIQUE: Given an undirected graph $G = (V,E)$ and an integer k, does G have a clique of size k? CLIQUE is W[1]-complete for the parameter k. Let $G = (V,E)$ and k be a given CLIQUE instance. For each $v \in V$, we define $S(v)$ to be a set containing v and all its neighbors. We construct an instance of PARTIAL-SET-MULTICOVER where $B = V$, $\mathscr{S} = \{S(v) \mid v \in V\}$, each set in $\mathscr{S}$ has cost 1, the budget is k, the covering requirement of each element of B is k, and the covering request is k. The reader can verify that this reduction is indeed correct.

The W[1]-membership proof uses a careful reduction to the SHORT-TURING-MACHINE-COMPUTATION problem [8]. □

COROLLARY 3.8. *Copeland*^{0}-EXTENSION-BRIBERY *is* W[1]-*complete for the parameter* (k, B), *where k is the difference in score between the preferred candidate and the current winner, and B is the budget, and where the only allowed extensions are to add the preferred candidate.*

However, there is some hope for the case where each voter ranks only few candidates (a situation corresponding to PARTIAL-SET-MULTICOVER instances where each set contains almost all members of the base set). The parametrized complexity of this variant of Copeland^0-EXTENSION-BRIBERY remains open (note that this also justifies why it was worthwhile to prove Theorem 3.4).

One can apply a similar analysis as for the case of Copeland^0 to the cases of Borda and maximin. However, there it turns out that EXTENSION-BRIBERY does not resemble PARTIAL-SET-MULTICOVER but rather SET-MULTICOVER. Thus, using the standard approximation algorithm for SET-MULTICOVER [22] we get the following result.

THEOREM 3.9. *For the cases of Borda*-EXTENSION-BRIBERY *(in the optimistic scoring model) and maximin*-EXTENSION-BRIBERY, *where the only allowed extensions are to add the preferred candidate, there are* $O(\log m)$-*approximation algorithms (where m is the number of candidates).*

The sizes of the base sets for the SET-MULTICOVER instances that we construct in the proof of the above theorem depend on (a) the number of candidates that have a higher score than the preferred candidate (for the case of Borda) or (b) the number of candidates "blocking" our preferred candidate's way to becoming the winner (for the case of maximin). If our preferred candidate is close to winning (in the sense of these candidate sets being small), we can try to solve the corresponding SET-MULTICOVER instance using a dynamic programming algorithm (whose running time would, nonetheless, be exponential, but with the exponent depending on the sizes of these candidate sets).

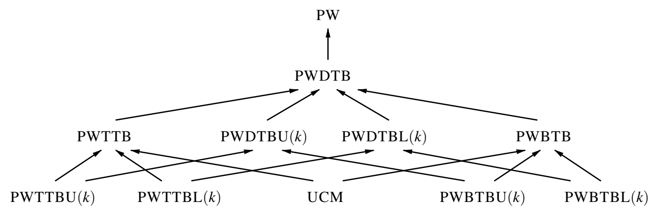
4. POSSIBLE WINNER PROBLEM

Given an election $E = (C, V)$, with the list V of votes being partial orders over the set of candidates C, the original POSSIBLE-WINNER problem (PW, for short) asks whether there is an extension of the given partial votes into complete ones over C such that a distinguished candidate wins the election. This problem was introduced by Konczak and Lang [25] and has received significant attention (see, e.g., [33, 3, 9, 35, 2]).

We consider the complexity of the possible winner problem if the partial votes have the special form of truncated ballots. We formulate these problems in the co-winner model, for a voting rule $\mathscr{R}$.

$\mathscr{R}$-POSSIBLE-WINNER-WITH-TOP-TRUNCATED-BALLOTS	
Given:	An election $E = (C, V)$, with possibly top-truncated ballots in V, and a distinguished candidate $p \in C$.
Question:	Can p be made an $\mathscr{R}$ winner of the election that results from E by fully extending all truncated ballots?

We define the corresponding problems for bottom- and doubly-truncated ballots analogously and abbreviate these three problems by, respectively, PWTTB, PWBTB, and PWDTB, omitting the voting rule $\mathscr{R}$. They capture the constructive variants as the possible winner problem does. Of course, one may also define the corresponding variants of the necessary winner problem.



$$\begin{tikzcd}
 & & & & \mathrm{PW} & & & & \\
 & & & & \mathrm{PWDTB} \arrow[u] & & & & \\
 & \mathrm{PWTTB} \arrow[urrr] & & \mathrm{PWDTBU}(k) \arrow[ur] & & \mathrm{PWDTBL}(k) \arrow[ul] & & \mathrm{PWBTB} \arrow[ulll] & \\
\mathrm{PWTTBU}(k) \arrow[ur] \arrow[urrr] & & \mathrm{PWTTBL}(k) \arrow[ul] \arrow[urrr] & & \mathrm{UCM} \arrow[ulll] \arrow[urrr] & & \mathrm{PWBTBU}(k) \arrow[ulll] \arrow[ur] & & \mathrm{PWBTBL}(k) \arrow[ulll] \arrow[ul]
\end{tikzcd}$$

Figure 1: A hierarchy of possible winner problems

PWTTB is closely related to, but different from, the decision problem EXTENSION-BRIBERY for the zero-cost model. While in PWTTB we have to extend all votes to linear orders, in EXTENSION-BRIBERY we have the freedom to extend votes only partially. Still, EXTENSION-BRIBERY reduces to PWTTB for each scoring rule in the optimistic scoring, zero-cost model.

Let k be a fixed positive constant. We consider the following restriction of PWTTB to *top-truncated ballots with upper-bounded (lower-bounded) length k*: For every top-truncated ballot B, the number of candidates ranked in B is at most k (at least k). The restriction of PWTTB to such ballots is denoted by PWTTBU(k) (PWTTBL(k)). Analogous definitions can be made for bottom-truncated and doubly-truncated ballots. Let the corresponding possible winner problems with truncated ballots of either upper- or lower-bounded length be denoted by PWBTBU(k), PWBTBL(k), PWDTBU(k), and PWDTBL(k). Truncated ballots with upper-bounded length may be seen as being heavily truncated, whereas truncated ballots with lower-bounded length may be seen as being only moderately truncated. Because UCM uses ballots that are either full or empty, neither PWTTBU(k) nor PWTTBL(k) is a more general problem than UCM. An analogous comment applies to PWBTBU(k), PWBTBL(k), PWDTBU(k), PWDTBL(k).

On the other hand, we immediately have from the definitions that UCM is a special case of both PWTTB and PWBTB, which in turn are special cases of PWDTB, and PWDTB is a special case of PW. This is stated in Proposition 4.1 and shown in Figure 1. In this figure, an arrow between two problems, $A \to B$, means that A (polynomial-time many-one) reduces to B.

PROPOSITION 4.1. *Among the possible winner problems with and without truncated ballots defined above, we have the reductions shown in Figure 1.*

Now, since the problems PWTTB, PWBTB, and PWDTB are sandwiched between PW and UCM (recall that UCM is a special case of PW), they immediately inherit any membership in P result from PW and any NP-hardness result from UCM. Thus, from known results about the PW and UCM problems for common voting rules [10, 25, 3, 33], we can classify these into three groups: (1) PW is in P for, e.g., plurality, veto, Condorcet, and plurality with runoff (in the co-winner case); (2) UCM is NP-hard for, e.g., Copeland, STV, maximin, ranked pairs, most scoring rules, plus all rules for which winner determination is NP-hard; (3) PW is NP-hard and UCM is in P for, e.g., Bucklin, voting trees, plurality with runoff (in the unique-winner case), and k-approval. Therefore, among the voting rules listed above, the complexity of PWTTB, PWBTB, and PWDTB is not yet known for Bucklin, voting trees, plurality with runoff (in the unique-winner case), and k-approval.

THEOREM 4.2. *For k-approval, the problems* PWDTB *and,* a fortiori, PWTTB *and* PWBTB *are in* P.

PROOF. Let $V = (v_1, \dots, v_n)$ be a given list of doubly-truncated ballots over a set C of m candidates, with the given values $\mathrm{top}(v_i)$

and bottom(v_i) for each voter v_i in V. Let $p \in C$ be the distinguished candidate. To decide whether p is a winner, we transform the given instance into the following network flow problem:

1. For each i, $1 \leq i \leq n$, if top$(v_i) < k$, and the position of p is not revealed in v_i (i.e., p is among the unranked candidates in v_i), then add p at position top$(v_i)+1$ in v_i. Let $V' = (v'_1, \dots, v'_n)$ be the corresponding modified profile with adjusted values top(v'_i), $1 \leq i \leq n$.
2. For each i, $1 \leq i \leq n$: (a) if top$(v'_i) \geq k$, then let Z_i be the set containing the first k candidates of v'_i; (b) if top$(v'_i) < k$ and bottom$(v'_i) \leq m-k$, then let Z_i be the set containing the first top(v'_i) candidates of v'_i; (c) if top$(v'_i) < k$ and bottom$(v'_i) > m-k$, then let Z_i be the set cotaining the first top(v'_i) candidates plus the first bottom$(v'_i) - m + k$ candidates which are ranked at the bottom of v'_i.
3. For each $c \in C$, let $S(c) = \|\{i \mid c \in Z_i\}\|$.
4. The flow network contains $n+m+1$ nodes: (a) one node c for each candidate $c \in C \setminus \{p\}$, (b) one node v'_i for each voter $v'_i \in V'$, (c) a source s, and (d) a sink t.
5. The flow network contains the following edges: (a) there is an edge from s to every $c \in C \setminus \{p\}$ with capacity $S(p)-S(c)$; (b) there is an edge from $c \in C \setminus \{p\}$ to $v'_i \in V'$ with capacity 1 if and only if the position of c is not revealed in v'_i; (c) there is an edge from every $v'_i \in V'$ to t with capacity

$$\begin{cases} 0 & \text{if top}(v'_i) \geq k; \\ k - \text{top}(v'_i) & \text{if top}(v'_i) < k \text{ and bottom}(v'_i) \leq m-k; \\ m - \text{top}(v'_i) & \\ \quad - \text{bottom}(v'_i) & \text{if top}(v'_i) < k \text{ and bottom}(v'_i) > m-k. \end{cases}$$

We claim that p is a possible winner in the k-approval election (C,V) if and only if there is a flow of value $\sum_{i=1}^{n} a_i$ in the network constructed above, where (1) $a_i = 0$ if top$(v'_i) \geq k$, (2) $a_i = k - $ top(v'_i) if top$(v'_i) < k$ and bottom$(v'_i) \leq m-k$, and (3) $a_i = m - $ top$(v'_i) - $ bottom(v'_i) if top$(v'_i) < k$ and bottom$(v'_i) > m-k$.

Assume that p is a possible k-approval winner for (C,V). That means that there is an extension of the list of truncated ballots V into a list W of complete ones such that p is a k-approval winner of election (C,W). Without loss of generality, we can assume that p is placed at the first possible position in each vote v_i where its position is unrevealed. Let (C,V') be the profile thus modified. The points every candidate gets in the profile (C,V') correspond to the values $S(c)$ of the above construction. We now show that there is a flow of value $\sum_{i=1}^{n} a_i$ in the network. First note that $\sum_{i=1}^{n} a_i$ is the sum of the unranked candidates among the first k positions in all votes. Since no candidate gets more points than p in (C,W), there is a flow of value at most $S(p)-S(c)$ from s to every node c for each candidate $c \in C \setminus \{p\}$. If in the list of complete ballots candidate c takes a position in a vote v_i that was unrevealed in V', there is a flow of value one from c to v_i. Further, from each node v_i, $1 \leq i \leq n$, there is a flow to the sink t whose value corresponds to the unrevealed candidates among the first k positions. Hence there is a flow of the desired value in this network.

Now assume that there is a flow of value $\sum_{i=1}^{n} a_i$ in the network. For the given election (C,V), we again first place candidate p at the first possible position in the votes where its position was unrevealed before, and we refer to the modified profile by V'. If there is a flow of value one from node c to v_i, candidate c is placed among the first k positions in vote v_i. The sum of all a_i ensures that all first k positions are taken in all votes, and the capacity of $S(p)-S(c)$ from the source s to the nodes corresponding to the candidates $c \in C \setminus \{p\}$ ensures that no candidate can get more points than p. Hence, completing the profile V' as described results in a k-approval election in which p is a winner. □

5. RELATED WORK

Although most of the literature in voting theory assumes that votes are full linear orders, truncated ballots have been considered in a few papers. Brams and Sanver [7] consider ballots that consist of ranked lists of approved candidates (i.e., of possibly truncated ballots), and they propose two specific ways of aggregating them, namely preference approval voting and fallback voting. Aggregating truncated ballots also appears in a stream of work in database theory, where aggregating ordered lists of results to queries (provided by web search engines, for example) corresponds to aggregating several ordered lists in which, typically, every element appears only in some lists, not in all (see, e.g., [12, 17, 1]). Assuming that the elements that are not ranked in a list are considered less relevant than the elements that appear in it, it is clear that the problem consists in aggregating a set of top-truncated rankings into a complete ranking. The aggregation rules used in this community are median rankings, which minimize some distance to the ballots, such as, typically, the Kemeny rule. One reason why these median ranking rules are used by this community is that their definition extends in a straightforward way to truncated rankings. Here we focus on other rules, and we also show that any other possible rule can be extended in a systematic way to truncated ballots, by considering all possible extensions of the truncated ballots. Rank aggregation with partial information is also important in peer-reviewing [30], and it received considerable attention from the machine learning community. For example, doubly-truncated votes are a special case of a similar notion studied by Lebanon and Mao [26]. Finally, Endriss et al. [16] provide a general approach to voting with many kinds of ballots, including truncated ballots.

Section 3 deals with the issue of campaign management for the case of truncated votes. Campaign management as a computational problem appeared, e.g., in [13, 31]. In particular, the latter work considered the problem SUPPORT-BRIBERY where the campaign manager can extend votes. However, there the voters already have fixed preference orders, whereas in EXTENSION-BRIBERY the manager has the freedom to ask the voters to additionally rank any subset of candidates in any order. In this sense EXTENSION-BRIBERY is similar to the BRIBERY problem [18], except that in BRIBERY if we buy a given vote, we can rewrite it completely.

Section 4 relates to computing possible winners. This notion has been introduced by Konczak and Lang [25] and has been studied from a computational point of view (see, e.g., [33, 3, 29]). The relation to preference elicitation is further explored by Walsh [32]. Parameterized complexity results for the possible winner problem are given by Betzler et al. [4]. Kalech et al. [24] design voting protocols in which agents submit their preferences incrementally, in rounds, and study experimentally the probability that there exists a necessary winner given the amount of information known. Lu and Boutilier [27] use minimax regret to output a robust winner given incomplete preferences, which is a way of coping with the possibly high number of possible winners. Xia and Conitzer [34] use the maximum likelihood approach to define voting rules form incomplete votes. Finally, a restricted variant of the possible winner problem, where the incompleteness of the profile comes from the fact that some alternatives were not known initially, is the *possible winner problem with respect to the addition of new alternatives* (PWNA) [9, 35, 2]. In PWNA, each voter has a full linear order over the set C of "known" candidates, but there is also a set A of additional candidates. The question is whether it is possible to extend the linear orders of the voters to rank all the candidates in $C \cup A$, so that some given candidate is a winner. At first sight it may look as if PWNA were a subproblem of PWTTB, PWBTB, and PWDTB. However, in the PWNA problem the remaining alternatives can be ranked anywhere, whereas in the case of truncated ballots they have to come between the ranked top and ranked bottom candidates.

6. CONCLUSIONS AND FUTURE WORK

We have studied various aspects of elections when the input votes are truncated rankings consisting of a ranked list of top candidates and/or a ranked list of bottom candidates. We proposed ways of extending some voting rules to truncated ballots, and have addressed the complexity of the problem of persuading voters to extend their votes so as to ensure that a given candidate wins. Considering truncated ballots as incompletely specified rankings, we showed that determining possible winners makes sense in this setting. While the complexity of the possible winner problem for truncated ballots follows from known results for many voting rules, we provided an efficient algorithm in one of the few other cases.

The generalization of voting rules to truncated ballots, and the study of such generalizations, deserves more attention. One goal is to extend common social-choice-theoretic properties to truncated ballots and to see if the properties fulfilled by a voting rule are still fulfilled in this generalization. Introducing new properties applicable only to (properly) truncated ballots may also make sense. Considering truncated ballots as incomplete ballots, how likely is it that the result is determined (i.e., that there exists a *necessary* winner) when all voters have specified ballots of a given size? (Kalech et al. [24] address a similar question in a different setting.) It is obvious how to prove that for the Bucklin rule: If top lists are of size $\lceil m/2 \rceil$ then there exists a necessary winner. It may also be interesting to build *elicitation protocols* so as to ask voters to expand their truncated ballots in a minimal way, in terms of the number of bits exchanged, so that the outcome of the vote is determined. Finally, we still have to address the complexity of PWDTB for plurality with runoff (in the unique-winner case), voting trees, and Bucklin.

Acknowledgements. We thank the referees for their many useful comments. This work was supported in part by DFG grant RO-1202/15-1, SFF grant "Cooperative Normsetting" of HHU Düsseldorf, a DAAD/MAEE/MESR grant for a project in the PROCOPE programme, AGH Univ. of Technology grant 11.11.120.865, the Foundation for Polish Science's Homing/Powroty program, and the ANR project ComSoc (ANR-09-BLAN-0305).

7. REFERENCES

[1] N. Ailon. Aggregation of partial rankings, p-ratings and top-m lists. *Algorithmica*, 57(2):284–300, 2010.

[2] D. Baumeister, M. Roos, and J. Rothe. Computational complexity of two variants of the possible winner problem. In *Proc. AAMAS-11*, pages 853–860, May 2011.

[3] N. Betzler and B. Dorn. Towards a dichotomy of finding possible winners in elections based on scoring rules. *Journal of Computer and System Sciences*, 76(8):812–836, 2010.

[4] N. Betzler, S. Hemmann, and R. Niedermeier. A multivariate complexity analysis of determining possible winners given incomplete votes. In *Proc. IJCAI-09*, pages 53–58, July 2009.

[5] N. Betzler, R. Niedermeier, and G. Woeginger. Unweighted coalitional manipulation under the Borda rule is NP-hard. In *Proc. IJCAI-11*, pages 55–60, July 2011.

[6] M. Bläser. Computing small partial coverings. *Information Processing Letters*, 85(6):327–331, 2003.

[7] S. Brams and R. Sanver. Voting systems that combine approval and preference. In S. Brams, W. Gehrlein, and F. Roberts, editors, *The Mathematics of Preference, Choice, and Order: Essays in Honor of Peter C. Fishburn*, pages 215–237. Springer, 2009.

[8] M. Cesati. The Turing way to parameterized complexity. *Journal of Computer and System Sciences*, 67(4):654–685, 2003.

[9] Y. Chevaleyre, J. Lang, N. Maudet, and J. Monnot. Possible winners when new candidates are added: The case of scoring rules. In *Proc. AAAI-10*, pages 762–767, July 2010.

[10] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate? *J.ACM*, 54(3):Article 14, 2007.

[11] J. Davies, G. Katsirelos, N. Narodytska, and T. Walsh. Complexity of and algorithms for Borda manipulation. In *Proc. AAAI-11*, pages 657–662, Aug. 2011.

[12] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proc. WWW-01*, pages 613–622, Mar. 2001.

[13] E. Elkind and P. Faliszewski. Approximation algorithms for campaign management. In *Proc. WINE-10*, pages 473–482, 2010.

[14] E. Elkind, P. Faliszewski, and A. Slinko. Cloning in elections: Finding the possible winners. *JAIR*, 42:529–573, 2011.

[15] P. Emerson. The original Borda count and partial voting. *Social Choice and Welfare*. To appear.

[16] U. Endriss, M. Pini, F. Rossi, and K. Venable. Preference aggregation over restricted ballot languages: Sincerity and strategy-proofness. In *Proc. IJCAI-09*, pages 122–127, July 2009.

[17] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists. *SIAM Journal on Discrete Mathematics*, 17(1):134–160, 2003.

[18] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. How hard is bribery in elections? *JAIR*, 35:485–532, 2009.

[19] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. Using complexity to protect elections. *C.ACM*, 53(11):74–82, 2010.

[20] P. Faliszewski, E. Hemaspaandra, and H. Schnoor. Manipulation of Copeland elections. In *Proc. AAMAS-10*, pages 367–374, 2010.

[21] P. Faliszewski and A. Procaccia. AI's war on manipulation: Are we winning? *AI Magazine*, 31(4):52–64, 2010.

[22] U. Feige. A threshold of $\ln n$ for approximating set cover. *J.ACM*, 45(4):634–652, 1998.

[23] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Anyone but him: The complexity of precluding an alternative. *Artificial Intelligence*, 171(5–6):255–285, 2007.

[24] M. Kalech, S. Kraus, G. Kaminka, and C. Goldman. Practical voting rules with partial information. *JAAMAS*, 22(1):151–182, 2011.

[25] K. Konczak and J. Lang. Voting procedures with incomplete preferences. In *Proc. Multidisciplinary IJCAI-05 Workshop on Advances in Preference Handling*, pages 124–129, July/Aug. 2005.

[26] G. Lebanon and Y. Mao. Non-parametric modeling of partially ranked data. *J. Machine Learning Research*, 9:2401–2429, 2008.

[27] T. Lu and C. Boutilier. Robust approximation and incremental elicitation in voting protocols. In *Proc. IJCAI-11*, pages 287–293, July 2011.

[28] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[29] M. Pini, F. Rossi, K. Venable, and T. Walsh. Possible and necessary winners in voting trees: Majority graphs vs. profiles. In *Proc. AAMAS-11*, pages 1–9, May 2011.

[30] M. Roos, J. Rothe, and B. Scheuermann. How to calibrate the scores of biased reviewers by quadratic programming. In *Proc. AAAI-11*, pages 255–260, Aug. 2011.

[31] I. Schlotter, P. Faliszewski, and E. Elkind. Campaign management under approval-driven voting rules. In *Proc. AAAI-11*, pages 726–731, Aug. 2011.

[32] T. Walsh. Uncertainty in preference elicitation and aggregation. In *Proc. AAAI-07*, pages 3–8, July 2007.

[33] L. Xia and V. Conitzer. Determining possible and necessary winners given partial orders. *JAIR*, 41:25–67, 2011.

[34] L. Xia and V. Conitzer. A maximum likelihood approach towards aggregating partial orders. In *Proc. IJCAI-11*, pages 446–451, July 2011.

[35] L. Xia, J. Lang, and J. Monnot. Possible winners when new alternatives join: New results coming up! In *Proc. AAMAS-11*, pages 829–836, May 2011.

Possible and Necessary Winners of Partial Tournaments

Haris Aziz
Institut für Informatik
TU München, Germany
aziz@in.tum.de

Markus Brill
Institut für Informatik
TU München, Germany
brill@in.tum.de

Felix Fischer
Statistical Laboratory
University of Cambridge, UK
fischerf@statslab.cam.ac.uk

Paul Harrenstein
Institut für Informatik
TU München, Germany
harrenst@in.tum.de

Jérôme Lang
LAMSADE, Université
Paris-Dauphine, France
lang@lamsade.dauphine.fr

Hans Georg Seedig
Institut für Informatik
TU München, Germany
seedigh@in.tum.de

ABSTRACT

We study the problem of computing possible and necessary winners for partially specified weighted and unweighted tournaments. This problem arises naturally in elections with incompletely specified votes, partially completed sports competitions, and more generally in any scenario where the outcome of some pairwise comparisons is not yet fully known. We specifically consider a number of well-known solution concepts—including the uncovered set, Borda, ranked pairs, and maximin—and show that for most of them possible and necessary winners can be identified in polynomial time. These positive algorithmic results stand in sharp contrast to earlier results concerning possible and necessary winners given partially specified preference profiles.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; J.4 [**Computer Applications**]: Social and Behavioral Sciences - Economics

General Terms

Economics, Theory, Algorithms

Keywords

Social Choice Theory, Tournament Solutions, Possible and Necessary Winners, Computational Complexity

1. INTRODUCTION

Many multi-agent situations can be modeled and analyzed using weighted or unweighted tournaments. Prime examples are voting scenarios in which pairwise comparisons between alternatives are decided by majority rule and sports competitions that are organized as round-robin tournaments. Other application areas include webpage and journal ranking, biology, psychology, and AI (also see [6], and the references therein). More generally, tournaments and tournament solutions are used as a mathematical tool for the analysis of all kinds of situations where a choice among a set of alternatives has to be made exclusively on the basis of pairwise comparisons.

When choosing from a tournament, relevant information may only be partly available. This could be because some preferences are yet to be elicited, some matches yet to be played, or certain comparisons yet to be made. In such cases, it is natural to speculate which are the potential and inevitable outcomes on the basis of the information already at hand.

For complete tournaments, a number of attractive solution concepts have been proposed (see, e.g., [6, 17]). Given any such solution concept S, *possible winners* of a partial tournament G are defined as alternatives that are selected by S in *some* completion of G, and *necessary winners* are alternatives that are selected in *all* completions. By a completion we here understand a complete tournament extending G.

In this paper we address the computational complexity of identifying the possible and necessary winners for a number of solution concepts whose winner determination problem for complete tournaments is tractable. We consider four of the most common tournament solutions—namely, Condorcet winners ($COND$), the Copeland solution (CO), the top cycle (TC), and the uncovered set (UC)—and three common solutions for weighted tournaments—Borda (BO), maximin (MM) and ranked pairs (RP). For each of these solution concepts, we characterize the complexity of the following problems: deciding whether a given alternative is a possible winner (PW), deciding whether a given alternative is a necessary winner (NW), and deciding whether a given subset of alternatives equals the set of winners in some completion (PWS). These problems can be challenging, as even unweighted partial tournaments may allow for an exponential number of completions. Our results are encouraging, in the sense that most of the problems can be solved in polynomial time. Table 1 summarizes our findings.

Similar problems have been considered before. For Condorcet winners, voting trees and the top cycle, it was already shown that possible and necessary winners are computable in polynomial time [16, 19, 20]. The same holds for computing possible Copeland winners that were considered in the context of sports tournaments [8].

A more specific setting that is frequently considered within the area of computational social choice differs from our setting in a subtle but important way that is worth being pointed out. There, tournaments are assumed to arise from pairwise majority comparisons on the basis of a profile

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

of individual voters' preferences.[1] Since a *partial* preference profile R need not conclusively settle every majority comparison, it may give rise to a *partial* tournament only. There are two natural ways to define possible and necessary winners for a partial preference profile R and solution concept S. The first is to consider the completions of the incomplete tournament $G(R)$ corresponding to R and the winners under S in these. This is covered by our more general setting. The second is to consider the completions of R and the winners under S in the corresponding tournaments.[2] Since every tournament corresponding to a completion of R is also a completion of $G(R)$ but not necessarily the other way round, the second definition gives rise to a *stronger* notion of a possible winner and a *weaker* notion of a necessary winner. Interestingly, and in sharp contrast to our results, determining these stronger possible and weaker necessary winners is computationally hard for many voting rules [16, 25].

In the context of this paper, we do not assume that tournaments arise from majority comparisons in voting or from any other specific procedure. This approach has a number of advantages. Firstly, it matches the diversity of settings to which tournament solutions are applicable, which goes well beyond social choice and voting. For instance, our results also apply to a question commonly encountered in sports competitions, namely, which teams can still win the cup and which future results this depends on (see, e.g., [8, 14]). Secondly, (partial) tournaments provide an informationally sustainable way of representing the relevant aspects of many situations while maintaining a workable level of abstraction and conciseness. For instance, in the social choice setting described above, the partial tournament induced by a partial preference profile is a much more succinct piece of information than the preference profile itself. Finally, specific settings may impose restrictions on the feasible extensions of partial tournaments. The positive algorithmic results in this paper can be used to efficiently approximate the sets of possible and necessary winners in such settings, where the corresponding problems may be intractable. The voting setting discussed above serves to illustrate this point.

2. PRELIMINARIES

A *partial tournament* is a pair $G = (V, E)$ where V is a finite set of alternatives and $E \subseteq V \times V$ an asymmetric relation on V, i.e., $(x, y) \in E$ implies $(y, x) \notin E$. If $(x, y) \in E$ we say that x *dominates* y. A *(complete) tournament* T is a partial tournament (V, E) for which E is also complete, i.e., either $(x, y) \in E$ or $(y, x) \in E$ for all distinct $x, y \in V$. We denote the class of complete tournaments by $\mathscr{T}$.

Let $G = (V, E)$ be a partial tournament. Another partial tournament $G' = (V', E')$ is called an *extension* of G, denoted $G \leq G'$, if $V = V'$ and $E \subseteq E'$. If E' is complete, G' is called a *completion of* G. We write $[G]$ for the set of completions of G, i.e., $[G] = \{T \in \mathscr{T} : G \leq T\}$.

For each $x \in V$, we define the *dominion of x in G* by

[1]See, e.g., [1, 2, 15, 24, 25] for the basic setting, [3] for parameterized complexity results, [12, 13] for probabilistic settings, and [7, 26] for settings with a variable set of alternatives.

[2]These two ways of defining possible and necessary winners are compared (both theoretically and experimentally) in [16, 20] for three solution concepts: Condorcet winners, voting trees and the top cycle.

S	PW_S		NW_S		PWS_S	
COND	in P	[16]	in P	[16]	in P	(Th. 1)
CO	in P	(Th. 2)[a]	in P	(Th. 2)[a]	in P	(Th. 2)
TC	in P	[16][a]	in P	[16]	in P	(Th. 3)
UC	in P	(Th. 4)	in P	(Th. 5)	NP-C	(Th. 6)
BO	in P	(Th. 7)[a]	in P	(Th. 9)	in P	(Th. 8)[b]
MM	in P	(Th. 10)[a]	in P	(Th. 11)	in P	(Th. 12)[b]
RP	NP-C	(Th. 13)	coNP-C	(Th. 14)	NP-C	(Cor. 1)

[a] This P-time result contrasts with the intractability of the same problem for partial preference profiles [16, 25].

[b] Assuming that the weight n is polynomial in the size of the partial tournament.

Table 1: Complexity of computing possible winners (PW) and necessary winners (NW) and of checking whether a given subset of alternatives is a possible winning set (PWS) under different solution concepts given partial tournaments.

$D^+_G(x) = \{y \in V : (x, y) \in E\}$, and the *dominators of x in G* by $D^-_G(x) = \{y \in V : (y, x) \in E\}$. For $X \subseteq V$, we let $D^+_G(X) = \bigcup_{x\in X} D^+_G(x)$ and $D^-_G(X) = \bigcup_{x\in X} D^-_G(x)$.

For given $G = (V, E)$ and $X \subseteq V$, we further write $E^{X\rightarrow}$ for the set of edges obtained from E by adding all missing edges from alternatives in X to alternatives not in X, i.e.,

$$E^{X\rightarrow} = E \cup \{(x, y) \in X \times V : y \notin X \text{ and } (y, x) \notin E\}.$$

We use $E^{X\leftarrow}$ as an abbreviation for $E^{V\setminus X\rightarrow}$, and respectively write $E^{x\rightarrow}$, $E^{x\leftarrow}$, $G^{X\rightarrow}$, and $G^{X\leftarrow}$ for $E^{\{x\}\rightarrow}$, $E^{\{x\}\leftarrow}$, $(V, E^{X\rightarrow})$, and $(V, E^{X\leftarrow})$.

Let n be a positive integer. A *partial n-weighted tournament* is a pair $G = (V, w)$ consisting of a finite set of alternatives V and a weight function $w : V \times V \to \{0, \ldots, n\}$ such that for each pair $(x, y) \in V \times V$ with $x \neq y$, $w(x, y) + w(y, x) \leq n$. We say that $T = (V, w)$ is a *(complete) n-weighted tournament* if for all $x, y \in V$ with $x \neq y$, $w(x, y) + w(y, x) = n$. A (partial or complete) *weighted* tournament is a (partial or complete) n-weighted tournament for some $n \in \mathbb{N}$. The class of n-weighted tournaments is denoted by $\mathscr{T}_n$. Observe that with each partial 1-weighted tournament (V, w) we can associate a partial tournament (V, E) by setting $E = \{(x, y) \in V : w(x, y) = 1\}$. Thus, (partial) n-weighted tournaments can be seen to generalize (partial) tournaments, and we may identify $\mathscr{T}_1$ with $\mathscr{T}$.

The notations $G \leq G'$ and $[G]$ can be extended naturally to partial n-weighted tournaments $G = (V, w)$ and $G' = (V', w')$ by letting $(V, w) \leq (V', w')$ if $V = V'$ and $w(x, y) \leq w'(x, y)$ for all $x, y \in V$, and $[G] = \{T \in \mathscr{T}_n : G \leq T\}$.

For given $G = (V, w)$ and $X \subseteq V$, we further define $w^{X\rightarrow}$ such that for all $x, y \in V$,

$$w^{X\rightarrow}(x, y) = \begin{cases} n - w(y, x) & \text{if } x \in X \text{ and } y \notin X, \\ w(x, y) & \text{otherwise,} \end{cases}$$

and set $w^{X\leftarrow} = w^{V\setminus X\rightarrow}$. Moreover, $w^{x\rightarrow}$, $w^{x\leftarrow}$, $G^{X\rightarrow}$, and $G^{X\leftarrow}$ are defined in the obvious way.

We use the term *solution concept* for functions S that associate with each (complete) tournament $T = (V, E)$, or with each (complete) weighted tournament $T = (V, w)$, a choice set $S(T) \subseteq V$. A solution concept S is called *resolute* if $|S(T)| = 1$ for each tournament T. In this paper we will consider the following solution concepts: *Condorcet winners*

(*COND*), *Copeland* (*CO*), *top cycle* (*TC*), and *uncovered set* (*UC*) for tournaments, and *maximin* (*MM*), *Borda* (*BO*), and *ranked pairs* (*RP*) for weighted tournaments. Of these only ranked pairs is resolute. Formal definitions will be provided later in the paper.

3. POSSIBLE & NECESSARY WINNERS

A solution concept selects alternatives from complete tournaments or complete weighted tournaments. A partial (weighted) tournament, on the other hand, can be extended to a number of complete (weighted) tournaments, and a solution concept selects a (potentially different) set of alternatives for each of them.

For a given a solution concept S, we can thus define the set of *possible winners* for a partial (weighted) tournament G as the set of alternatives selected by S from *some* completion of G, i.e., as $PW_S(G) = \bigcup_{T\in[G]} S(T)$. Analogously, the set of *necessary winners* of G is the set of alternatives selected by S from *every* completion of G, i.e., $NW_S(G) = \bigcap_{T\in[G]} S(T)$. We can finally write $PWS_S(G) = \{S(T) : T \in [G]\}$ for the set of *sets* of alternatives that S selects for the different completions of G.

Note that $NW_S(G)$ may be empty even if S selects a non-empty set of alternatives for each tournament $T \in [G]$, and that $|PWS_S(G)|$ may be exponential in the number of alternatives of G. It is also easily verified that $G \leq G'$ implies $PW_S(G') \subseteq PW_S(G)$ and $NW_S(G) \subseteq NW_S(G')$, and that $PW_S(G) = \bigcup_{G\leq G'} NW_S(G')$ and $NW_S(G) = \bigcap_{G\leq G'} PW_S(G')$.

Deciding membership in the sets $PW_S(G)$, $NW_S(G)$, and $PWS_S(G)$ for a given solution concept S and a partial (weighted) tournament G is a natural computational problem. We will respectively refer to these problems as PW_S, NW_S, and PWS_S, and will study them for the solution concepts mentioned at the end of the previous section.[3]

For complete tournaments T we have $[T] = \{T\}$ and thus $PW_S(T) = NW_S(T) = S(T)$ and $PWS_S(T) = \{S(T)\}$. As a consequence, for solution concepts S with an NP-hard winner determination problem—like *Banks*, *Slater*, and *TEQ*—the problems PW_S, NW_S, and PWS_S are NP-hard as well. We therefore restrict our attention to solution concepts for which winners can be computed in polynomial time.

For irresolute solution concepts, PWS_S may appear a more complex problem than PW_S. We are, however, not aware of a polynomial-time reduction from PW_S to PWS_S. The relationship between these problems may also be of interest for the "classic" possible winner setting with partial preference profiles.

4. UNWEIGHTED TOURNAMENTS

In this section, we consider the following well-known solution concepts for unweighted tournaments: Condorcet winners, Copeland, top cycle, and uncovered set. Weighted tournaments will then be considered in Section 5.

4.1 Condorcet Winners

Condorcet winners are a very simple solution concept and will provide a nice warm-up. An alternative $x \in V$ is a *Condorcet winner* of a complete tournament $T = (V, E)$ if it dominates all other alternatives, i.e., if $(x, y) \in E$ for all $y \in V \setminus \{x\}$. The set of Condorcet winners of tournament T will be denoted by $COND(T)$; obviously this set is always either a singleton or empty.

It is readily appreciated that the possible Condorcet winners of a partial tournament $G = (V, E)$ are precisely the undominated alternatives, and that a necessary Condorcet winner of G should already dominate all other alternatives. Both properties can be verified in polynomial time.

Each of the sets in $PWS_{COND}(G)$ is either a singleton or the empty set, and determining membership for a singleton is obviously tractable. Checking whether $\emptyset \in PWS_{COND}(G)$ is not quite that simple. First observe that $\emptyset \in PWS_{COND}(G)$ if and only if there is an extension G' of G in which every alternative is dominated by some other alternative. Given a particular $G = (V, E)$, we can define an extension $G' = (V, E')$ of G by iteratively adding edges from dominated alternatives to undominated ones until this is no longer possible. Formally, let

$$E_0 = E \text{ and } E_{i+1} = E_i \cup \{(x, y) \in X_i \times Y_i : (y, x) \notin E_i\},$$

where X_i and Y_i denote the dominated and undominated alternatives of (V, E_i), respectively. Finally define $E' = \bigcup_{i=0}^{|V|} E_i$, and observe that this set can be computed in polynomial time.

Now, for every undominated alternative x of G' and every dominated alternative y of G', we not only have $(x, y) \in E'$, but also $(x, y) \in E$. This is the case because in the inductive definition of E' only edges from dominated to undominated alternatives are added in every step. It is therefore easily verified that $PWS_{COND}(G)$ contains $\emptyset$ if and only if the set of undominated alternatives in G' is either empty or is of size three or more. We have shown the following easy result.

THEOREM 1. *PW_{COND}, NW_{COND}, and PWS_{COND} can be solved in polynomial time.*

The results for PW_{COND} and NW_{COND} also follow from Proposition 2 of Lang et al. [16] and Corollary 2 of Konczak and Lang [15]. We further note that Theorem 1 is a corollary of corresponding results for maximin in Section 5.2. The reason is that a Condorcet winner is the maximin winner of a 1-weighted tournament, and a tournament does not admit a Condorcet winner if and only if all alternatives are maximin winners.

4.2 Copeland

Copeland's solution selects alternatives based on the number of other alternatives they dominate. Define the *Copeland score* of an alternative x in tournament $T = (V, E)$ as $s_{CO}(x, T) = |D_T^+(x)|$. The set $CO(T)$ then consists of all alternatives that have maximal Copeland score. Since Copeland scores coincide with Borda scores in the case of 1-weighted tournaments, the following is a direct corollary of the results in Section 5.1.

THEOREM 2. *NW_{CO}, PW_{CO}, and PWS_{CO} can be solved in polynomial time.*

PW_{CO} can alternatively be solved via a polynomial-time reduction to maximum network flow (see, e.g., [8], p. 51).

[3]Formally, the input for each of the problems consists of an encoding of the partial (n-weighted) tournament G and, for partial n-weighted tournaments, the number n.

4.3 Top Cycle

A subset $X \subseteq V$ of alternatives in a (partial or complete) tournament (V, E) is *dominant* if every alternative in X dominates every alternative outside X. The *top cycle* of a tournament $T = (V, E)$, denoted by $TC(T)$, is the unique *minimal* dominant subset of V.

Lang et al. have shown that possible and necessary winners for TC can be computed efficiently by greedy algorithms ([16], Corollaries 1 and 2). For PWS_{TC}, we not only have to check that there exists a completion such that the set in question is dominating, but also that there is no smaller dominating set. It turns out that this can still be done in polynomial time.

Theorem 3. *PWS_{TC} can be solved in polynomial time.*

Proof Sketch. Consider a partial tournament $G = (V, E)$ and a set $X \subseteq V$ of alternatives. If X is a singleton, the problem reduces to checking whether $X \in PWS_{COND}(G)$. If X is of size two or if one of its elements is dominated by an outside alternative, $X \notin PWS_{TC}(G)$. Therefore, we can without loss of generality assume that $|X| \geq 3$ and $(y, x) \notin E$ for all $y \in V \setminus X$ and $x \in X$. The *Smith set* of a partial tournament is defined as the minimal dominant subset of alternatives [22].[4] It can be shown that there exists a completion $T \in [G]$ with $TC(T) = X$ if and only if the Smith set of the partial tournament $(X, E|_{X \times X})$ equals the whole set X. Since Brandt et al. [4] have shown that the Smith set of a partial tournament can be computed efficiently, the theorem follows. $\square$

4.4 Uncovered Set

Given a tournament $T = (V, E)$, an alternative $x \in V$ is said to *cover* another alternative $y \in V$ if $D^+_T(y) \subseteq D^+_T(x)$, i.e., if every alternative dominated by y is also dominated by x. The *uncovered set* of T, denoted by $UC(T)$, then is the set of alternatives that are not covered by some other alternative. A useful alternative characterization of the uncovered set is via the *two-step principle*: an alternative is in the uncovered set if and only if it can reach every other alternative in at most two steps.[5] Formally, $x \in UC(T)$ if and only if for all $y \in V \setminus \{x\}$, either $(x, y) \in E$ or there is some $z \in V$ with $(x, z), (z, y) \in E$. We denote the two-step dominion $D^+_E(D^+_E(x))$ of an alternative x by $D^{++}_E(x)$.

We first consider PW_{UC}, for which we check for each alternative whether it can be reinforced to reach every other alternative in at most two steps.

Theorem 4. *PW_{UC} can be solved in polynomial time.*

Proof. For a given partial tournament $G = (V, E)$ and an alternative $x \in V$, we check whether x is in $UC(T)$ for some completion $T \in [G]$.

Consider the graph $G' = (V, E'')$ where E'' is derived from E as follows. First, we let $D^+(x)$ grow as much as possible by letting $E' = E^{x\rightarrow}$. Then, we do the same for its two-step dominion by defining E'' as $E'^{D^+_{E'}(x)\rightarrow}$. Now it can be shown that $x \in PW_{UC}(G)$ if and only if $V = \{x\} \cup D^+_{E''}(x) \cup D^{++}_{E''}(x)$. $\square$

A similar argument yields the following.

Theorem 5. *NW_{UC} can be solved in polynomial time.*

Proof. For a given partial tournament $G = (V, E)$ and an alternative $x \in V$, we check whether x is in $UC(T)$ for all completions $T \in [G]$.

Consider the graph $G' = (V, E'')$ with E'' defined as follows. First, let $E' = E^{x\leftarrow}$. Then, expand it to $E'' = E'^{D^-_{E'}(x)\rightarrow}$. Intuitively, this makes it as hard as possible for x to beat alternatives outside of its dominion in two steps. Then it can be shown that $x \in NW_{UC}(G)$ if and only if $V = \{x\} \cup D^+_{E''}(x) \cup D^{++}_{E''}(x)$. $\square$

For all solution concepts considered so far—Condorcet winners, Copeland, and top cycle—PW and PWS have the same complexity. One might wonder whether a result like this holds more generally, and whether there could be a polynomial-time reduction from PWS to PW. The following result shows that this is not the case, unless P=NP.

Theorem 6. *PWS_{UC} is NP-complete.*

Proof Sketch. Let $G = (V, E)$ be a partial tournament. Given a set $X \subseteq V$ and a completion $T \in [G]$, it can be checked in polynomial time whether $X = UC(T)$. Hence, PWS_{UC} is obviously in NP.

NP-hardness can be shown by a reduction from Sat. For each Boolean formula φ in conjunctive normal-form with a set C of clauses and set P of propositional variables, we construct a partial tournament $G_\varphi = (V_\varphi, E_\varphi)$. Define

$$V_\varphi = C \times \{0, 1\} \cup P \times \{0, \ldots, 5\} \cup \{0, 1, 2\},$$

i.e., along with three auxiliary alternatives, we introduce for each clause two alternatives and for each propositional variable six. We write c_i, p_i, C_i, and P_i for (c, i), (p, i), $\{c_i : c \in C\}$, and $\{p_i : p \in P\}$, respectively. Let

$$X = C \times \{0\} \cup P \times \{0, 1, 2\} \cup \{0, 1, 2\}.$$

Then, E_φ is defined such that it contains no edges between alternatives in $V_\varphi \setminus X$. For alternatives $x \in X$, E_φ is given by the following table, in which each line is of the form $D^-_{G_\varphi}(x) \cap V \setminus X \to x \to D^+_{G_\varphi}(x) \cap X$ and where it is understood that x dominates all alternatives in $V_\varphi \setminus X$ unless specified otherwise. For improved readability some curly braces have been omitted and a comma indicates set-theoretic union.

$$\begin{aligned}
\{p_3 : p \in c\}, \{p_4 : \bar{p} \in c\}, c_1 \to c_0 &\to 2, P_2, \{p_1 : p \notin c\}, \{p_0 : \bar{p} \notin c\} \\
p_3 \to p_0 &\to 0, p_2, \{c_0 : \bar{p} \in c\} \\
p_4 \to p_1 &\to 0, p_2, \{c_0 : p \in c\} \\
P_3, P_4, p_5 \to p_2 &\to 2, \{q_0, q_1 : q \neq p\} \\
P_3, P_4 \to \ 0 \ &\to 2, C_0, P_2 \\
C_1, P_5 \to \ 1 \ &\to 0, C_0, P_2 \\
\emptyset \to \ 2 \ &\to 1, P_0, P_1
\end{aligned}$$

It now suffices to show that E_φ is specified in such a way that X is the uncovered set of some completion of G_φ if and only if φ is satisfiable.

For every $p \in P$, the edges between p_0, p_1, and 1 are left unspecified. The idea is that p_0 and p_1 are the only candidates to cover p_5, p_0 and 1 are the only candidates to cover p_4, and p_1 and 1 are the only candidates to cover p_3. As $p_0 \in D^+_{G_\varphi}(p_3)$, $p_1 \in D^+_{G_\varphi}(p_4)$, and $1 \in D^+_{G_\varphi}(p_5)$, there are two possibilities of extending G_φ in such a way that p_3, p_4 and p_5 are covered simultaneously and X is the uncovered set. Either all the edges in

[4]For complete tournaments, the Smith set coincides with the top cycle.

[5]In graph theory, vertices satisfying this property are often called *kings*.

(a) $\{(p_0, p_1), (p_1, 1), (1, p_0)\}$, or all those in

(b) $\{(p_1, p_0), (p_0, 1), (1, p_1)\}$

have to be added to E_φ to achieve this (additionally some edges among $V_\varphi \setminus X$ have to be set appropriately as well). Possibility (a) corresponds to setting p to "true." In this case, p_1 also covers c_1 for every clause $c \in C$ that contains p. Possibility (b) corresponds to setting p to "false" and causes p_0 to cover c_1 for every clause $c \in C$ that contains $\bar{p}$. Moreover, for each $c \in C$, the only candidates in X to cover c_1 are p_1 if $p \in c$ and p_0 if $\bar{p} \in c$. Observe that $1 \in D^+_{G_\varphi}(c_1)$ for all $c \in C$. Thus, if $\bar{p} \in c$, p_1 covering p_3 precludes p_0 covering c_1. Similarly, if $p \in c$, p_0 covering p_4 precludes p_1 covering c_1. Accordingly, if T is a completion of G_φ in which X is the uncovered set, one can read off a valuation satisfying φ from how the edges between p_0, p_1, and 1 are set in T. For the opposite direction, a satisfying valuation for φ is a recipe for extending G_φ to a tournament in which X is the uncovered set. It can be checked that every alternative in X reaches every other alternative in at most two steps, whereas every alternative in $V_\varphi \setminus X$ is covered by some alternative in X. $\square$

5. WEIGHTED TOURNAMENTS

We now turn to weighted tournaments, and in particular consider the solution concepts Borda, maximin, and ranked pairs.

5.1 Borda

The Borda solution (BO) is typically used in a voting context, where it is construed as based on voters' rankings of the alternatives: each alternative receives $|V| - 1$ points for each time it is ranked first, $|V| - 2$ points for each time it is ranked second, and so forth; the solution concept then chooses the alternatives with the highest total number of points. In the more general setting of weighted tournaments, the *Borda score* of alternative $x \in V$ in $G = (V, w)$ is defined as $s_{BO}(x, G) = \sum_{y \in V \setminus \{x\}} w(x, y)$ and the *Borda winners* are the alternatives with the highest Borda score. If $w(x, y)$ represents the number of voters that rank x higher than y, the two definitions are equivalent.

Before we proceed further, we define the notion of a b-matching, which will be used in the proofs of two of our results. Let $H = (V_H, E_H)$ be an undirected graph with vertex capacities $b : V_H \to \mathbb{N}_0$. Then, a *b-matching* of H is a function $m : E_H \to \mathbb{N}_0$ such that for all $v \in V_H$, $\sum_{e \in \{e' \in E_H : v \in e'\}} m(e) \leq b(v)$. The *size* of b-matching m is defined as $\sum_{e \in E_H} m(e)$. It is easy to see that if $b(v) = 1$ for all $v \in V_H$, then a maximum size b-matching is equivalent to a maximum cardinality matching. In a b-matching problem with upper *and* lower bounds, there further is a function $a : V_H \to \mathbb{N}_0$. A feasible b-matching then is a function $m : E_H \to \mathbb{N}_0$ such that $a(v) \leq \sum_{e \in \{e' \in E_H : v \in e'\}} m(e) \leq b(v)$.

If H is bipartite, then the problem of computing a maximum size feasible b-matching with lower and upper bounds can be solved in strongly polynomial time ([21], Chapter 21). We will use this fact to show that PW_{BO} and PWS_{BO} can both be solved in polynomial time. While the following result for PW_{BO} can be shown using Theorem 6.1 of [14], we give a direct proof that can then be extended to PWS_{BO}.

THEOREM 7. *PW_{BO} can be solved in polynomial time.*

PROOF SKETCH. Let $G = (V, w)$ be a partial n-weighted tournament, $x \in V$. We give a polynomial-time algorithm for checking whether $x \in PW_{BO}(G)$, via a reduction to the problem of computing a maximum size b-matching of a bipartite graph.

Let $G^{x\to} = (V, w^{x\to})$ denote the graph obtained from G by maximally reinforcing x, and $s^* = s_{BO}(x, G^{x\to})$ the Borda score of x in $G^{x\to}$. From $G^{x\to}$, we then construct a bipartite graph $H = (V_H, E_H)$ with vertices $V_H = V \setminus \{x\} \cup E^{<n}$, where $E^{<n} = \{\{i, j\} \subseteq V \setminus \{x\} : w(i, j) + w(j, i) < n\}$,[6] and edges $E_H = \{\{v, e\} : v \in V \setminus \{x\} \text{ and } v \in e \in E^{<n}\}$. We further define vertex capacities $b : V_H \to \mathbb{N}_0$ such that $b(\{i, j\}) = n - w(i, j) - w(j, i)$ for $\{i, j\} \in E^{<n}$ and $b(v) = s^* - s_{BO}(v, G^{x\to})$ for $v \in V \setminus \{x\}$.

Now observe that in any completion $T = (V, w') \in [G^{x\to}]$, $w'(i, j) + w'(j, i) = n$ for all $i, j \in V$ with $i \neq j$. The sum of the Borda scores in T is therefore $n|V|(|V| - 1)/2$. Some of the weight has already been used up in $G^{x\to}$; the weight which has not yet been used up is equal to $\alpha = n|V|(|V| - 1)/2 - \sum_{v \in V} s_{BO}(v, G^{x\to})$. We claim that $x \in PW_{BO}(G)$ if and only if H has a b-matching of size at least α.

Since H can be constructed efficiently, and since a maximum size b-matching can be computed in strongly polynomial time, our algorithm runs in polynomial time. $\square$

We now extend this proof to a pseudo-polynomial time algorithm for PWS_{BO}.

THEOREM 8. *PWS_{BO} can be solved in pseudo-polynomial time.*

PROOF SKETCH. Let $G = (V, w)$ be a partial n-weighted tournament, and $X \subseteq V$. We give a pseudo-polynomial time algorithm for checking whether $X \in PWS_{BO}(G)$, via a reduction to the problem of computing a maximum b-matching of a graph with lower and upper bounds.

Assume that there is a target Borda score s^* and a completion $T \in [G]$ with $X \in PWS_{BO}(T)$ and $s_{BO}(x, T) = s^*$ for all $x \in X$. Then, the maximum Borda score of an alternative not in X is $s^* - 1$. As we do not know s^* in advance, we initialize it to the maximum possible Borda score of $n(|V| - 1)$ and decrease it until we find a completion that makes X the set of Borda winners or until $s^* = 0$.

For a given s^*, we construct a bipartite graph $H = (V_H, E_H)$ with vertices $V_H = V \cup E^{<n}$, where $E^{<n} = \{\{i, j\} \subseteq V : i \neq j, w(i, j) + w(j, i) < n\}$, and edges $E_H = \{\{v, e\} : v \in V \text{ and } v \in e \in E^{<n}\}$. Lower bounds $b : V_H \to \mathbb{N}_0$ and upper bounds $a : V_H \to \mathbb{N}_0$ are defined as follows: For vertices $x \in X$, lower and upper bounds coincide and are given by $a(x) = b(x) = s^* - s_{BO}(x, G)$. All other vertices $v \in V_H \setminus X$ have a lower bound of $a(v) = 0$. Upper bounds for these vertices are defined such that $b(v) = s^* - s_{BO}(v, G) - 1$ for $v \in V \setminus X$, and $b(\{i, j\}) = n - w(i, j) - w(j, i)$ for $\{i, j\} \in E^{<n}$.

Observe that the weight not yet used up in G is equal to $\alpha = n|V|(|V| - 1)/2 - \sum_{v \in V} s_{BO}(v, G)$. We claim that membership of X in $PWS_{BO}(G)$ can be decided via the following algorithm. Start by initializing s^* to $n(|V| - 1)$. In each step, construct the bipartite graph H described above for the current value of s^*. If H has a feasible b-matching of size at least α, return "yes." Otherwise decrement s^* by one and repeat. If $s^* = 0$ and no feasible b-matching of size at least α has been found, return "no."

[6]Note that $w(i, j) = w^{x\to}(i, j)$ for alternatives $i, j \in V \setminus \{x\}$.

It is straightforward to prove that this algorithm is correct. The essential idea is that a feasible b-matching of size α corresponds to a completion of G in which all alternatives in X have the same Borda score s^*, while all other alternatives have a strictly smaller Borda score.

The algorithm requires at most $n(|V|-1)$ iterations, each of which involves the computation of a maximum size b-matching of a bipartite graph H. The latter can be done in strongly polynomial time, so the algorithm has an overall running time of $O(n \cdot |V|^k)$ for some constant k. □

We conclude this section by showing that NW_{BO} can be solved in polynomial time as well.

Theorem 9. *NW_{BO} can be solved in polynomial time.*

Proof. Let $G = (V, w)$ be a partial weighted tournament, $x \in V$. We give a polynomial-time algorithm for checking whether $x \in NW_{BO}(G)$.

Let $G' = G^{x\leftarrow}$. We want to check whether some other alternative $y \in V \setminus \{x\}$ can achieve a Borda score of more than $s^* = s_{BO}(x, G')$. This can be done separately for each $y \in V \setminus \{x\}$ by reinforcing it as much as possible in G'. If for some y, $s_{BO}(y, G'^{y\rightarrow}) > s^*$, then $x \notin NW_{BO}(G)$. If, on the other hand, $s_{BO}(y, G'^{y\rightarrow}) \leq s^*$ for all $y \in V \setminus \{x\}$, then $x \in NW_{BO}(G)$. □

Since the Borda and Copeland solutions coincide in unweighted tournaments, the above results imply that PW_{CO} and NW_{CO} can be solved in polynomial time. The same is true for PWS_{CO}, because the Copeland score is bounded by $|V| - 1$.

5.2 Maximin

The *maximin score* $s_{MM}(x, T)$ of an alternative x in a weighted tournament $T = (V, w)$, is given by its worst pairwise comparison, i.e., $s_{MM}(x, T) = \min_{y \in V \setminus \{x\}} w(x, y)$. The *maximin solution*, also known as *Simpson's method* and denoted by MM, returns the set of all alternatives with the highest maximin score.

We first show that PW_{MM} is polynomial-time solvable by reducing it to the problem of finding a maximum cardinality matching of a graph.

Theorem 10. *PW_{MM} can be solved in polynomial time.*

Proof Sketch. We show how to check whether $x \in PW_{MM}(G)$ for a partial n-weighted tournament $G = (V, w)$. Consider the graph $G^{x\rightarrow} = (V, w^{x\rightarrow})$. Then, $s_{MM}(x, G^{x\rightarrow})$ is the best possible maximin score x can get among all completions of G. If $s_{MM}(x, G^{x\rightarrow}) \geq \frac{n}{2}$, then we have $s_{MM}(y, T) \leq w^{x\rightarrow}(y, x) \leq \frac{n}{2}$ for every $y \in V \setminus \{x\}$ and every completion $T \in [G^{x\rightarrow}]$ and therefore $x \in PW_{MM}(G)$. Now consider $s_{MM}(x, G^{x\rightarrow}) < \frac{n}{2}$. We will reduce the problem of checking whether $x \in PW_{MM}(G)$ to that of finding a maximum cardinality matching, which is known to be solvable in polynomial time [11]. We want to find a completion $T \in [G^{x\rightarrow}]$ such that $s_{MM}(x, T) \geq s_{MM}(y, T)$ for all $y \in V \setminus \{x\}$. If there exists a $y \in V \setminus \{x\}$ such that $s_{MM}(x, G^{x\rightarrow}) < s_{MM}(y, G^{x\rightarrow})$, then we already know that $x \notin PW_{MM}(G)$. Otherwise, each $y \in V \setminus \{x\}$ derives its maximin score from at least one particular edge (y, z) where $z \in V \setminus \{x, y\}$ and $w(y, z) \leq s_{MM}(x, G^{x\rightarrow})$. Moreover, it is clear that in any completion, y and z cannot both achieve a maximin score of less than $s_{MM}(x, G^{x\rightarrow})$ from edges (y, z) and (z, y) at the same time.

Construct the following undirected and unweighted graph $H = (V_H, E_H)$ where $V_H = V \setminus \{x\} \cup \{\{i, j\} \subseteq V : i \neq j\}$. Build up E_H such that: $\{i, \{i, j\}\} \in E_H$ if and only if $i \neq j$ and $w^{x\rightarrow}(i, j) \leq s_{MM}(x, G^{x\rightarrow})$. In this way, if i is matched to $\{i, j\}$ in H, then i derives a maximin score of less than or equal to $s_{MM}(x, G^{x\rightarrow})$ from his comparison with j. Clearly, H is polynomial in the size of G. Then, the claim is that $x \in PW_{MM}(G)$ if and only if there exists a matching of cardinality $|V| - 1$ in H. □

For NW_{MM} we apply a similar technique as for NW_{BO}: to see whether $x \in NW_{MM}(G)$, we start from the graph $G^{x\leftarrow}$ and check whether some other alternative can achieve a higher maximin score than x in a completion of $G^{x\leftarrow}$.

Theorem 11. *NW_{MM} can be solved in polynomial time.*

We conclude the section by showing that PWS_{MM} can be solved in pseudo-polynomial time. The proof proceeds by identifying the maximin values that could potentially be achieved simultaneously by all elements of the set in question, and solving the problem for each of these values using similar techniques as in the proof of Theorem 10.

Theorem 12. *PWS_{MM} can be solved in pseudo-polynomial time.*

Proof Sketch. Let $G = (V, w)$ be a partial n-weighted tournament, and $X \subseteq V$. We give a pseudo-polynomial time algorithm for checking whether $X \in PWS_{MM}(G)$.

If $X \in PWS_{MM}(G)$ there must be a completion $T \in [G]$ and $s^* \in \{0, \ldots, n\}$ such that $s_{MM}(i, T) = s^*$ for all $i \in X$. We check for each possible s^* whether X can be made the set of maximin winners with a maximin score of s^*.

Assume that $s^* > \frac{n}{2}$. Then, $X \in PWS_{MM}$ if and only if X is a singleton $\{x\}$ and $w^{x\rightarrow}(x, j) > \frac{n}{2}$ for all $j \in V \setminus \{x\}$.

Let $s^* < \frac{n}{2}$. Similarly as in the proof of Theorem 10, we construct an undirected unweighted graph $H = (V_H, E_H)$ with $V_H = V \cup \{\{i, j\} \subseteq V : i \neq j\}$ and capacity function c. Build up E_H such that if $i \in X$ then $\{i, \{i, j\}\} \in E_H$ if and only if $w(i, j) \leq s^* \leq n - w(j, i)$, and if $i \in V \setminus X$ then $\{i, \{i, j\}\} \in E_H$ if and only if $w(i, j) < s^*$. We claim that there is a matching of cardinality $|V|$ in H if and only if there is a completion T in which for all $i \in X$, $s_{MM}(i, T) = s^*$ and for all $i \in V \setminus X$, $s_{MM}(i, T) < s^*$. Intuitively speaking, an edge $\{i, \{i, j\}\}$ in such a matching corresponds to $w(i, j) = s^*$ in the completion if $i \in X$ and to $w(i, j) < s^*$ if $x \in V \setminus X$.

Finally, we study separately the case $s^* = \frac{n}{2}$. The difference with the case $s^* < \frac{n}{2}$ is that now, it is possible that both (i, j) and (j, i) account for the maximin score of i and j in the completion. We create a flow network $N = (V_N, E_N, s, t, c)$ where $V_N = V_H \cup \{s, t\}$. For each $i \in V$, there is an edge (s, i) in E_N with capacity 1. For all distinct $i, j \in V$, there are two edges $(i, \{i, j\})$ and $(j, \{i, j\})$ in E_N with capacity 1 if $w(i, j) \leq s^* \leq n - w(j, i)$; otherwise there are no edges between i, j and $\{i, j\}$ in N. For all $i, j \in X$, there is an edge $(\{i, j\}, t)$ in E_N with capacity 2. For each $i \in V$ and each $j \in V \setminus X$, E_N contains an edge $(\{i, j\}, t)$ with capacity 1. We claim that the maximum value of the flow equals $|V|$ if and only if $X \in PWS_{MM}(G)$. Here, an edge $(i, \{i, j\})$ with nonzero flow in a maximum flow corresponds to $w(i, j) = w(j, i) = s^*$ in the completion if $i, j \in X$ and to $w(i, j) < s^*$ if $i \in V \setminus X$.

Obviously, all cases can be completed in pseudo-polynomial time. □

5.3 Ranked Pairs

The method of *ranked pairs* (*RP*) is the only resolute solution concept considered in this paper. Given a weighted tournament $T = (V, w)$, it returns the unique undominated alternative of a transitive tournament T' on V constructed in the following manner. First order the (directed) edges of T in decreasing order of weight, breaking ties according to some exogenously given tie-breaking rule. Then consider the edges one by one according to this ordering. If the current edge can be added to T' without creating a cycle, then do so; otherwise discard the edge.[7]

It is readily appreciated that this procedure, and thus the winner determination problem for RP, is computationally tractable. The possible winner problem, on the other hand, turns out to be NP-hard. This also shows that tractability of the winner determination problem, while necessary for tractability of PW, is not generally sufficient.

Theorem 13. *PW_{RP} is NP-complete.*

Proof Sketch. Membership in NP is obvious, as for a given completion and a given tie-breaking rule, the ranked pairs winner can be found efficiently.

NP-hardness can be shown by a reduction from Sat. For a Boolean formula φ in conjunctive normal-form with a set C of clauses and set P of propositional variables, we construct a partial 8-weighted tournament $G_\varphi = (V_\varphi, w_\varphi)$ as follows. For each variable $p \in P$, V_φ contains two *literal alternatives* p and $\bar{p}$ and two *auxiliary alternatives* p' and $\bar{p}'$. For each clause $c \in C$, there is an alternative c. Finally, there is an alternative d for which membership in $PW_{RP}(G_\varphi)$ is to be decided.

In order to conveniently describe the weight function w_φ, let us introduce the following terminology. For two alternatives $x, y \in V_\varphi$, say that there is a *heavy* edge from x to y if $w_\varphi(x, y) = 8$ (and therefore $w_\varphi(y, x) = 0$). A *medium* edge from x to y means $w_\varphi(x, y) = 6$ and $w_\varphi(y, x) = 2$, and a *light* edge from x to y means $w_\varphi(x, y) = 5$ and $w_\varphi(y, x) = 3$. Finally, a *partial* edge between x and y means $w_\varphi(x, y) = w_\varphi(y, x) = 1$.

We are now ready to define w_φ. For each variable $p \in P$, we have heavy edges from p to $\bar{p}'$ and from $\bar{p}$ to p', and partial edges between p and p' and between $\bar{p}$ and $\bar{p}'$. For each clause $c \in C$, we have a medium edge from c to d and a heavy edge from the literal alternative $\ell_i \in \{p, \bar{p}\}$ to c if the corresponding literal ℓ_i appears in the clause c. Finally, we have heavy edges from d to all auxiliary alternatives and light edges from d to all literal alternatives. For all pairs x, y for which no edge has been specified, we define $w_\varphi(x, y) = w_\varphi(y, x) = 4$.

Observe that the only pairs of alternatives for which w_φ is not fully specified are those pairs that are connected by a partial edge. It can be shown that alternative d is a possible ranked pairs winner in G_φ if and only if φ is satisfiable. Intuitively, choosing a completion w' of w_φ such that $w'(p', p)$ is large and $w'(\bar{p}', \bar{p})$ is small corresponds to setting the variable p to "true." □

[7]The variant of ranked pairs originally proposed by Tideman [23], which was also used by Xia and Conitzer [25], instead chooses a set of alternatives, containing any alternative that is selected by the above procedure for *some* way of breaking ties among edges with equal weight. We do not consider this irresolute version of ranked pairs because it was recently shown that winner determination for this variant is NP-hard [5]. As mentioned in Section 3, this immediately implies that all problems concerning possible or necessary winners are NP-hard as well.

Since the ranked pairs method is resolute, hardness of PWS_{RP} follows immediately.

Corollary 1. *PWS_{RP} is NP-complete.*

Computing necessary ranked pairs winners turns out to be coNP-complete. This is again somewhat surprising, as computing necessary winners is often considerably easier than computing possible winners, both for partial tournaments and partial preference profiles [25].

Theorem 14. *NW_{RP} is coNP-complete.*

Proof Sketch. Membership in coNP is again obvious. For hardness, we give a reduction from UnSat that is a slight variation of the reduction in the proof of Theorem 13. We introduce a new alternative d^*, which has heavy edges to all alternatives in V_φ except d. Furthermore, there is a light edge from d to d^*. It can be shown that d^* is a necessary ranked pairs winner in this partial 8-weighted tournament if and only if φ is unsatisfiable. □

6. DISCUSSION

The problem of computing possible and necessary winners for partial preference profiles has recently received a lot of attention. In this paper, we have investigated this problem in a setting where partially specified (weighted or unweighted) *tournaments* instead of profiles are given as input. We have summarized our findings in Table 1.

A key conclusion is that computational problems for partial tournaments can be significantly easier than their counterparts for partial profiles. For example, possible Borda or maximin winners can be found efficiently for partial tournaments, whereas the corresponding problems for partial profiles are NP-complete [25].

While tractability of the winner determination problem is necessary for tractability of the possible or necessary winners problems, the results for ranked pairs in Section 5.3 show that it is not sufficient. We further considered the problem of deciding whether a given subset of alternatives equals the winner set for some completion of the partial tournament. The results for the uncovered set in Section 4.4 imply that this problem cannot be reduced to the computation of possible or necessary winners, but whether a reduction exists in the opposite direction remains an open problem.

Partial tournaments have also been studied in their own right, independent of their possible completions. For instance, Peris and Subiza [18] and Dutta and Laslier [10] have generalized several tournament solutions to incomplete tournaments by directly adapting their definitions. In this context, the notion of possible winners suggests a canonical way to generalize a tournament solution to incomplete tournaments. The positive computational results in this paper are an indication that this may be a promising approach.

Other open problems follow more directly from our results. For example, it will be interesting to see whether strongly polynomial-time algorithms exist for PWS_{BO} and PWS_{MM}. Furthermore, we have not examined the complexity of computing possible and necessary winners for some attractive tournament solutions such as the minimal covering set, the bipartisan set [17] and weighted versions of the top cycle and the uncovered set [9].

An interesting related question that goes beyond the computation of possible and necessary winners is the following: when the winners are not yet fully determined, which unknown comparisons need to be learned, or which matches should be played? The construction of a policy tree defining an optimal protocol minimizing the number of questions to be asked or the number of matches to be played, in the worst case or on average, is an even more challenging issue that we leave for further research.

ACKNOWLEDGEMENTS

This material is based on work supported by the Deutsche Forschungsgemeinschaft under grants BR 2312/9-1, BR 2312/10-1, and FI 1664/1-1. Jérôme Lang has been supported by the ANR project ComSoc (ANR-09-BLAN-0305).

REFERENCES

[1] D. Baumeister and J. Rothe. Taking the final step to a full dichotomy of the possible winner problem in pure scoring rules. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI)*, pages 1019–1020, 2010.

[2] N. Betzler and B. Dorn. Towards a dichotomy for the possible winner problem in elections based on scoring rules. *Journal of Computer and System Sciences*, 76(8): 812–836, 2010.

[3] N. Betzler, S. Hemmann, and R. Niedermeier. A multivariate complexity analysis of determining possible winners given incomplete votes. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 53–58, 2009.

[4] F. Brandt, F. Fischer, and P. Harrenstein. The computational complexity of choice sets. *Mathematical Logic Quarterly*, 55(4):444–459, 2009.

[5] M. Brill and F. Fischer. The price of neutrality for the ranked pairs method. Unpublished manuscript, 2012.

[6] I. Charon and O. Hudry. A survey on the linear ordering problem for weighted or unweighted tournaments. *4OR*, 5(1):5–60, 2007.

[7] Y. Chevaleyre, J. Lang, N. Maudet, and J. Monnot. Possible winners when new candidates are added: The case of scoring rules. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, pages 762–767. AAAI Press, 2010.

[8] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization.* Wiley and Sons, 1998.

[9] P. De Donder, M. Le Breton, and M. Truchon. Choosing from a weighted tournament. *Mathematical Social Sciences*, 40(1):85–109, 2000.

[10] B. Dutta and J.-F. Laslier. Comparison functions and choice correspondences. *Social Choice and Welfare*, 16 (4):513–532, 1999.

[11] J. Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[12] N. Hazon, Y. Aumann, S. Kraus, and M. Wooldridge. Evaluation of election outcomes under uncertainty. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 959–966, 2008.

[13] M. Kalech, S. Kraus, G. A. Kaminka, and C. V. Goldman. Practical voting rules with partial information. *Proceedings of the 10th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 22(1):151–182, 2011.

[14] W. Kern and D. Paulusma. The computational complexity of the elimination problem in generalized sports competitions. *Discrete Optimization*, 1(2):205–214, 2004.

[15] K. Konczak and J. Lang. Voting procedures with incomplete preferences. In *Proceedings of the Multidisciplinary Workshop on Advances in Preference Handling*, pages 214–129. 2005.

[16] J. Lang, M. S. Pini, F. Rossi, D. Salvagnin, K. B. Venable, and T. Walsh. Winner determination in voting trees with incomplete preferences and weighted votes. *Journal of Autonomous Agents and Multi-Agent Systems*, 25(1):130–157, 2012.

[17] J.-F. Laslier. *Tournament Solutions and Majority Voting.* Springer-Verlag, 1997.

[18] J. E. Peris and B. Subiza. Condorcet choice correspondences for weak tournaments. *Social Choice and Welfare*, 16(2):217–231, 1999.

[19] M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Dealing with incomplete agents' preferences and an uncertain agenda in group decision making via sequential majority voting. In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 571–578. AAAI Press, 2008.

[20] M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Possible and necessary winners in voting trees: Majority graphs vs. profiles. In *Proceedings of the 10th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 311–318, 2011.

[21] A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency.* Springer, 2003.

[22] J. H. Smith. Aggregation of preferences with variable electorate. *Econometrica*, 41(6):1027–1041, 1973.

[23] T. N. Tideman. Independence of clones as a criterion for voting rules. *Social Choice and Welfare*, 4(3):185–206, 1987.

[24] T. Walsh. Uncertainty in preference elicitation and aggregation. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI)*, pages 3–8. AAAI Press, 2007.

[25] L. Xia and V. Conitzer. Determining possible and necessary winners under common voting rules given partial orders. *Journal of Artificial Intelligence Research*, 41: 25–67, 2011.

[26] L. Xia, J. Lang, and J. Monnot. Possible winners when new alternatives join: New results coming up! In *Proceedings of the 10th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 829–836, 2011.

Communication Complexity of Approximating Voting Rules[*]

Travis C. Service
Department of Electrical Engineering and Computer Science
Vanderbilt University
travis.c.service@vanderbilt.edu

Julie A. Adams
Department of Electrical Engineering and Computer Science
Vanderbilt University
juile.a.adams@vanderbilt.edu

ABSTRACT

This paper considers the communication complexity of approximating common voting rules. Both upper and lower bounds are presented. For n voters and m alternatives, it is shown that for all $\epsilon \in (0, 1)$, the communication complexity of obtaining a $1 - \epsilon$ approximation to Borda is $O(\log(\frac{1}{\epsilon})nm)$. A lower bound of $\Omega(nm)$ is provided for fixed small values of ϵ. The communication complexity of computing the true Borda winner is $\Omega(nm \log(m))$ [5]. Thus, in the case of Borda, one can obtain arbitrarily good approximations with less communication overhead than is required to compute the true Borda winner.

For other voting rules, no such $1 \pm \epsilon$ approximation scheme exists. In particular, it is shown that the communication complexity of computing any constant factor approximation, ρ, to Bucklin is $\Omega(\frac{nm}{\rho^2})$. Conitzer and Sandholm [5] show that the communication complexity of computing the true Bucklin winner is $O(nm)$. However, we show that for all $\delta \in (0, 1)$, the communication complexity of computing a m^δ approximate winner in Bucklin elections is $O(nm^{1-\delta} \log(m))$. For $\delta \in (\frac{1}{2}, 1)$, a lower bound of $\Omega(nm^{1-2\delta})$ is also provided.

Similar lower bounds are presented on the communication complexity of computing approximate winners in Copeland elections.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Theory

Keywords

voting, communication complexity, approximation

1. INTRODUCTION

The naive application of a voting rule requires each voter to send their entire preference order over the set of alternatives. If there are n voters and m alternatives communicating every voter's preference order requires $\Theta(nm \log(m))$ bits be communicated. However, using an intelligent elicitation protocol can result in significant savings. Conitzer and Sandholm [5] show that a number of voting rules can be computed with low communication overhead. For example, the communication complexity of plurality is $O(n \log(m))$ and the communication complexity of single transferable vote is $O(n \log^2(m))$. Conitzer and Sandholm further characterized the communication complexity of a number of common voting rules by presenting lower bounds on the communication complexity of each.

Not every voting rule can be computed with a low amount of communication overhead [5]. Determining the true winner in Borda and Copeland elections requires communication complexity $\Theta(nm \log(m))$. Likewise, computing the winner in a Bucklin election requires communication complexity $\Theta(nm)$.

Conitzer and Sandholm [5] suggest that when, for example, voting over issues of relatively low importance:

> Knowing which voting rules require little communication is especially important when the issue to be voted on is of low enough importance that the following is true: the parties involved are willing to accept a rule that tends to produce outcomes that are slightly less representative of the voters' preferences, if this rule reduces the communication burden on the voters significantly.

A more natural approach in such situations is to use a low communication complexity approximation to the desired voting rule. For example, rather than selecting plurality (with communication complexity $\Theta(n \log(m))$) over the preferred voting rule Borda (with communication complexity $\Theta(nm \log(m))$), it is more natural to obtain a $1 - \epsilon$ approximation to Borda, for some $\epsilon \in (0, 1)$, using a reduced amount of communication.

This paper considers the communication complexity of approximating common score-based rules. It is shown that it is possible to obtain arbitrarily good approximations to some voting rules using less communication than is required to compute the actual winner. For example, an approximation scheme for Borda voting is presented that, for all $\epsilon \in (0, 1)$, obtains a $1 - \epsilon$ approximation to Borda with communication complexity $O(\log(\frac{1}{\epsilon})nm)$. It is shown that, up to constant factors, this approximation scheme is optimal. That is, it is shown that for all $\delta \in (0, 1 - \frac{1}{\sqrt{2}})$, the communication

[*]This research has been supported by an ONR DEPSCOR Award # N000140911161.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

complexity of computing a $\frac{1}{\sqrt{2}} + \delta$ approximation to Borda is $\Omega\left(\frac{\delta^3}{\log(1/\delta)} nm - \log(\log(1/\delta))\right)$.

While some voting rules, such as Borda, admit $1 \pm \epsilon$ approximation schemes, others do not. We show that for any $\rho > 1$, the communication complexity of computing a ρ approximate winner in Bucklin elections is $\Omega\left(\frac{nm}{\rho^2}\right)$. Conitzer and Sandholm [5] show that $O(nm)$ bits of communication are sufficient to determine the true Bucklin winner.

For sufficiently good approximations, lower bounds on the communication complexity of $\Omega(nm)$ for computing approximate winners in Copeland elections are presented as well.

Both incremental preference elicitation [6, 8, 9] and approximation of voting rules [2, 3, 10] has seen increasing attention. Most work on approximation of voting rules considers rules under which it is NP-hard to compute the true winner [2, 3, 10], or attempts to employ approximation in an effort to guarantee strategy-proofness [11]. However, to the best of the authors' knowledge, this paper is the first to provide bounds on the communication complexity of computing approximate winners under voting rules.

The remainder of this paper is structured as follows. Preliminary definitions and background are presented in Section 2. Section 3 presents the upper and lower bounds on the communication complexity of approximating Borda, Bucklin, and Copeland. Section 4 presents a discussion of the results and some future work.

2. PRELIMINARIES

This section provides basic background related to voting rules, communication complexity, and the proof methods employed in this paper.

2.1 Voting Rules

Let V be a set of n voters and let A be a set of m alternatives. Each voter, v_i, has a strict preference order, $\succ_i$, over the m alternatives. A preference profile is a vector of voter preference orders.

A voting rule is a mapping from preference profiles to winning alternatives. In the interest of obtaining results on approximating voting rules, this paper restricts its attention to rules that assign each alternative a score-based on the voters' preference. That is, if $sc : A \rightarrow \mathbb{R}$ is a function assigning a score to each alternative, then the winner is the alternative that maximizes/minimizes its score. Score-based voting rules allow for natural measures of approximation.

This paper considers the following voting rules.

1. Borda: Each alternative a is awarded $m - k$ points for every voter that ranks a in its k-th position. The Borda winner is the alternative with the greatest score. If w is the Borda winner and a any other alternative, then the approximation ratio obtained by a is $\frac{sc(a)}{sc(w)} \leq 1$.

2. Bucklin: The Bucklin score of each alternative a is the minimum value of k such that a strict majority of voters rank a in one of the top k positions. The Bucklin winner is the alternative with the least score. If w is the Bucklin winner and a any other alternative, then the approximation ratio obtained by a is $\frac{sc(a)}{sc(w)} \geq 1$.

3. Copeland: An alternative a is said to defeat an alternative b in a pairwise election if a strict majority of voters prefer a to b. Under Copeland every alternative a receives one point for every alternative that a defeats in a pairwise election and half a point for every alternative a ties. The Copeland winner is the alternative with the greatest score. If w is the Copeland winner and a any other alternative, then the approximation ratio obtained by a is $\frac{sc(a)}{sc(w)} \leq 1$.

For Borda and Copeland, the approximation ratio obtained by a communication protocol, f, is $\rho \in [0, 1]$ if for every preference profile, P, $\frac{sc(f(P))}{sc(w)} \leq \rho$, where $f(P)$ is the alternative selected by f under P and w is the winning alternative in P. Likewise, f obtains a $\rho \geq 1$ approximation in Bucklin elections if for every preference profile, P, $\frac{sc(f(P))}{sc(w)} \geq \rho$.

For each voter $v \in V$ and alternative $a \in A$, let $v(a)$ be the rank of a in v's preference order. For example, if v has the preference order

$$x \succ y \succ a \succ z,$$

then $v(a) = 3$.

2.2 Communication Complexity

This paper employs the standard model of communication complexity [5, 7, 12]. The objective is to compute the outcome of a voting rule $f(\succ_1, \cdots, \succ_n)$. However, each piece of the input $\succ_i$ is known only to a single voter v_i. A protocol for computing f consists of a number of rounds. During each round, a single voter announces a single bit to all other voters. In a deterministic protocol, the next voter to announce a bit and the bit to be announced are completely determined by the preceding rounds and that voter's preference order. A communication pattern is the sequence of bits announced. The winner elected by the voters is then a function of the particular communication pattern observed. Note that in a k round deterministic protocol there are at most 2^k possible communication patterns.

During each round of the protocol all voters have observed the same sequence of communicated bits. The protocol terminates when sufficient information has been communicated for every voter to compute f, or in our case to determine an approximate winner. The communication complexity of approximating a voting rule f is the worst case number of bits sent by the best approximation protocol.

In order to compute lower bounds on the amount of communication required to approximate certain voting rules, a slight generalization of the standard lower bound technique of constructing a *fooling set* is employed. The definition for rules that select alternatives that maximize their score is presented. The definition for rules that select alternatives that minimize their score is analogous.

Definition 1 (Fooling set). *Let sc be a score-based voting rule. A ρ-fooling set S for sc is a set of preference profiles with the following properties:*

1. *$w \in A$ is the winning alternative in every preference profile in S under the score-based voting rule sc.*

2. *In every preference profile $P \in S$, every $a \in A \setminus \{w\}$ does not obtain a ρ-approximate solution in P. That is $\frac{sc(a)}{sc(w)} < \rho$, for every $a \in A \setminus \{w\}$.*

3. *For every N preference profiles $P_1 = (v_1^1, \cdots, v_n^1), \cdots, P_N = (v_1^N, \cdots, v_n^N)$ in S, there exists a vector of indices $(r_1, \cdots, r_n) \in \{1, \cdots, N\}^n$ such that w does not obtain a ρ-approximation in $P_r = (v_1^{r_1}, \cdots, v_n^{r_n})$ (i.e., it is possible to mix voters from the N preference profiles such that w no longer obtains a ρ-approximation). That is, there exists some alternative $a \in A$ such that $\frac{sc(w)}{sc(a)} < \rho$.*

Theorem 1. *If sc is a score-based voting rule and S a fooling set, then the deterministic communication complexity of computing a ρ-approximation to sc is $\Omega\left(\log\left(\frac{|S|}{N}\right)\right)$.*

Proof. Suppose there is a deterministic protocol D that computes a ρ-approximation to *sc* in $\left\lfloor \log\left(\frac{|S|}{N}\right)\right\rfloor - 1$ bits. Thus, there are at most $2^{\left\lfloor \log\left(\frac{|S|}{N}\right)\right\rfloor - 1} < \frac{|S|}{N}$ possible communication patterns. By the pigeonhole principle, there exists N preference profiles $P_i = (v_1^i, \cdots, v_n^i)$ for $i = 1, \cdots, N$ in S that have the same communication pattern under D.

Since all the P_i's exhibit the same communication pattern, the protocol must select the same alternative w under each. Let $r = (r_1, \cdots, r_n) \in \{i, j\}^n$ such that w does not obtain a ρ-approximation in P_r. By assumption, such an r exists. It is known that D produces the same communication pattern on P_r as it does on all the P_i's [1]. Since D exhibits the same communication pattern on P_r as it does all the P_i's, D must select w as the winning alternative in P_r. However, w does not obtain a ρ-approximation in P_r; a contradiction. Hence, the communication complexity of obtaining a ρ approximation is $\Omega\left(\log\left(\frac{|S|}{N}\right)\right)$. □

A natural question is whether Conitzer and Sandholm's [5] fooling set constructions already provide lower bounds on the communication complexity of computing approximate winners for the score-based voting rules considered in this paper. However, on inspection, it is observed that Conitzer and Sandholm's lower bound proofs for Borda, Bucklin, and Copeland construct fooling sets in which the single winning alternative a has a constant number of points more than the next highest alternative(s). Unfortunately, in our setting, Contizer and Sandholm's constructions are not strong enough to lower bound the deterministic communication requirements of approximately computing these rules. That is, the ratio of the scores of the winning alternative to the scores of the other alternatives need to be bounded away from 1. With Contizer and Sandholm's constructions, the ratio of the score of any alternative to that of the winning alternative tends towards 1 for increasingly large elections.

2.3 Probabilistic Method

The stronger requirements on the fooling sets needed to lower bound the deterministic communication complexity of approximately computing voting rules complicates the construction of fooling sets. However, the fooling set need not actually be constructed. Showing the *existence* of such a set is sufficient for the lower bound proofs.

Instead of explicit constructions, the lower bound proofs in this paper employ a powerful method from combinatorics to show the existence of objects satisfying certain properties. The probabilistic method proves the existence of a combinatorial object satisfying certain properties as follows:

1. First, construct a probability distribution over the objects of interest.

2. Second, show that an object drawn from that distribution possesses the desired properties with strictly positive probability.

Since with probability greater than zero the object drawn from the distribution satisfies the requirements, it is assured to exist. Using the probabilistic method, an appropriate fooling set can be shown to exist without providing an explicit construction.

The probabilistic method is employed in our lower bound proofs by constructing a distribution over preference profiles. This distribution over preference profiles then implicitly defines a distribution over sets of preference profiles (i.e., potential fooling sets). It is shown that a set of k preference profiles drawn from this distribution satisfies the fooling set properties with strictly positive probability. It can be concluded that a fooling set of size k exists, which implies a $\Omega(\log(\frac{k}{N}))$ lower bound on the communication complexity.

All of the presented lower bound results hinge on Chernoff bounds [4].

Theorem 2 (Chernoff [4])**.** *Let $X_1, \cdots, X_n$ be n independent random variables taking on values 0 or 1, such that $Pr(X_i = 1) = p$, for each $i = 1, \cdots, n$. Let $X = \sum_{i=1}^n X_i$ and let $\delta \in (0, 1)$ then*

$$Pr(X < (1-\delta)\mathbb{E}(X)) = Pr(X < (1-\delta)pn) < e^{\frac{-pn\delta^2}{2}}.$$

3. RESULTS

Upper bounds on the communication complexity of obtaining approximations to Borda and Bucklin are presented first. It is shown that arbitrarily good approximations to Borda can be obtained with less communication overhead than computing the true Borda winner. For Bucklin, it is shown that a number of non-constant approximations can be achieved with less communication complexity than computing the true Bucklin winner.

A number of lower bounds are then presented on the communication complexity of obtaining a number of approximation ratios with respect to Borda, Bucklin, and Copeland. In particular, it is shown that the Borda and Copeland voting rules require $\Omega(nm)$ communication complexity to compute sufficiently good constant factor approximations. For Bucklin, it is shown that for any constant ρ, the communication complexity of computing a *rho*-approximate winner in Bucklin elections is $\Omega(\frac{1}{\rho^2}nm)$, which, for fixed ρ, matches the upper bound given by Conitzer and Sandholm for computing the true Bucklin winner.

3.1 Upper Bounds

Conitzer and Sandholm [5] show that the communication complexity of determining the Borda winner is $\Theta(nm\log(m))$. Theorem 3 shows that a $(1-\epsilon)$ approximation to Borda can be obtained by a protocol with communication complexity $O(\log(\frac{1}{\epsilon})nm)$.

Informally, in the protocol presented in Theorem 3, each voter announces an approximate rank of each alternative in its preference order using a $O(\log(\frac{1}{\epsilon}))$ bits. That is, the preference order of each voter is divided into k equally sized segments and each voter indicates which of the k segments each alternative falls into using only $O(\log(k))$ bits per alternative. With this information, upper and lower bounds can be inferred for the Borda score of each alternative. The alternative with the greatest upper bound is selected as the winner. The proof of Theorem 3 shows that arbitrarily good approximation ratios can be obtained given an appropriate choice of k.

Theorem 3. *For all $\epsilon \in (0,1)$, there is a deterministic communication protocol that approximates Borda to within a factor of $1-\epsilon$ with communication complexity $O(\log(\frac{1}{\epsilon})nm)$.*

Proof. Given $\epsilon \in (0,1)$, let $k = \lceil \frac{4}{\epsilon} \rceil$. Let $a_1, a_2, \cdots, a_m$ be a fixed ordering on the alternatives in A. For each a_i, every voter, v, announces in order the value $l \in (0, k-1)$ such that

$$v(a_i) \in \left[\left\lceil \frac{lm}{k} \right\rceil, \left\lceil \frac{(l+1)m}{k} \right\rceil + 1 \right].$$

This procedure requires $\lceil \log(k) \rceil nm = O(\log(\frac{1}{\epsilon})nm)$ bits for fixed ϵ.

For a voter v and alternative a, let $v^l(a)$ be the value of l returned by voter v for a. For each alternative $a \in A$, define the following lower and upper bounds, $lb(a)$ and $ub(a)$, respectively, on a's true Borda score:

$$\begin{aligned} lb(a) &= \sum_{v \in V} \left(m - \left\lceil \frac{(v^l(a)+1)m}{k} \right\rceil + 1 \right), \\ ub(a) &= \sum_{v \in V} \left(m - \left\lceil \frac{v^l(a)m}{k} \right\rceil \right). \end{aligned}$$

In essence, the upper bound on a's true Borda score is obtained by assuming that a's true rank in each voter's preference order falls at the lower end of the range of ranks reported by each voter. The lower bound is obtained by assuming that a's true rank falls at the upper end of the range reported by each voter.

Then, for each $a \in A$

$$\begin{aligned} ub(a) - lb(a) &= \sum_{v \in V} \left(\left\lceil \frac{(v^l(a)+1)m}{k} \right\rceil - \left\lceil \frac{v^l(a)m}{k} \right\rceil - 1 \right) \\ &\leq \sum_{v \in V} \left(\frac{(v^l(a)+1)m}{k} + 1 - \frac{v^l(a)m}{k} - 1 \right) \\ &= \frac{nm}{k}. \end{aligned}$$

The protocol selects the alternative a with the greatest $ub(a)$ value. Now it is shown that a is a $1-\epsilon$ approximate winner.

Let w be the true Borda winner. Since the sum of all the alternatives scores is $\frac{nm(m-1)}{2}$, $sc(w) \geq \frac{n(m-1)}{2}$. Also since a was selected, $sc(w) \leq ub(w) \leq ub(a)$. The approximation ratio is

$$\begin{aligned} \frac{sc(a)}{sc(w)} &\geq \frac{lb(a)}{sc(w)} \\ &\geq \frac{ub(a) - \frac{nm}{k}}{sc(w)} \\ &= \frac{ub(a)}{sc(w)} - \frac{\frac{nm}{k}}{sc(w)} \\ &\geq \frac{ub(a)}{ub(a)} - \frac{2m}{k(m-1)} \\ &\geq 1 - \frac{4}{k} \\ &\geq 1 - \epsilon. \end{aligned}$$

□

It will be shown that obtaining any constant factor approximation to Bucklin requires $\Omega(nm)$ communication complexity. Since, the true Bucklin winner can be determined using $O(nm)$ bits of communication complexity, there does not exists any asymptotically better communication protocol to obtain a constant factor approximation to Bucklin. However, non-constant factor approximations can be easily obtained in Bucklin elections.

Theorem 4. *For every $\delta \in (0,1)$, there is a deterministic communication protocol that obtains a m^δ approximation to Bucklin with communication complexity $O(nm^{(1-\delta)}\log(m))$.*

Proof. Consider the following protocol. Every voter broadcasts the top $m^{(1-\delta)} - 1$ entries in its preference order using $O(nm^{(1-\delta)}\log(m))$ bits. If any alternative appears in the top $m^{(1-\delta)}$ positions in a strict majority of voters, then the true Bucklin winner must also appear in the top $m^{(1-\delta)}$ positions by a strict majority of the voters as well. Hence, the true Bucklin winner can be computed given the partial lists of preferences reported by each voter.

Otherwise, if no alternative appears in the top $m^{(1-\delta)}$ positions in a strict majority of voters, the Bucklin score of any alternative is at least $m^{(1-\delta)}$. Since the Bucklin score of every alternative is at most m, every alternative is a $\frac{m}{m^{(1-\delta)}} = m^\delta$ approximation solution. Hence, an arbitrary alternative may be selected in this case. □

3.2 Lower Bounds

All of the lower bound proofs will employ the same parameterized distribution over preference profiles.

Let the set of m alternatives be $A = \{a_1, \cdots, a_m\}$. Fix $w \in A$ and partition $A \setminus \{w\}$ into the following sets

1. $X = \{x_i : i = 1, \cdots, |X|\}$,
2. $Y = \{y_i^r : i = 1, \cdots, \frac{|Y|}{2} \text{ and } r \in \{0,1\}\}$
3. $Z = \{z_i^r : i = 1, \cdots, \frac{|Z|}{2} \text{ and } r \in \{0,1\}\}$.

The sizes of the sets X, Y, and Z will depend upon the particular voting rule and desired approximation ratio. Informally, we construct a distribution over preference profiles that satisfies the following properties:

1. w is preferred to every alternative in X by every voter,
2. for every alternative in $a \in Z \cup Y$, half of the voters prefer w to a and the other half prefer a to w, and

3. for every two alternatives $a, b \in A \setminus \{w\}$, half of the voters prefer a to b and the other half prefer b to a.

Thus, every alternative in $A \setminus \{w\}$ obtains a roughly average score under the considered voting rules.

We then show that for any N preference profiles drawn from this distribution with high probability it is possible to mix and match the voters from the N preference profiles such that, there is some alternative $z \in Z$ that an overwhelming majority of the voters prefer to all members of $Y \cup X$. That is, it is possible to mix and match the voters such that the score of some $z \in Z$ is significantly higher than z's score in any of the individual preference profiles. In particular, z's score will be significantly higher than that of w.

For $r \in \{0, 1\}$, let $Y_r = \{y_i^r : i = 1, \cdots, \frac{|Y|}{2}\}$, similarly for Z_r. Let π_X be a fixed permutation over X. Let $n = 2n'$.

Construct a distribution $\mathcal{P}$ over preference profiles as follows. Every preference profile from $\mathcal{P}$ is constructed using the following random procedure. For each $i \in \{1, \cdots, n'\}$, select $r \in \{0,1\}^{|Z_0|}$ and $j \in \{0,1\}^{|Y_0|}$ uniformly at random. Voter v_{2i} has preference order:

$$\begin{aligned}
&z_1^{r_1} \succ \cdots \succ z_{|Z_0|}^{r_{|Z_0|}} \\
&\succ y_1^{j_1} \succ \cdots \succ y_{|Y_0|}^{j_{|Y_0|}} \\
&\succ w \succ \pi_X(1) \succ \cdots \succ \pi_X(|X|) \\
&\succ y_1^{\neg j_1} \succ \cdots \succ y_{|Y_0|}^{\neg j_{|Y_0|}} \\
&\succ z_{|Z_0|}^{\neg r_{|Z_0|}} \succ \cdots \succ z_1^{\neg r_1}
\end{aligned}$$

Voter v_{2i-1} has preference order:

$$\begin{aligned}
&z_1^{\neg r_1} \succ \cdots \succ z_{|Z_0|}^{\neg r_{|Z_0|}} \\
&\succ y_1^{\neg j_1} \succ \cdots \succ y_{|Y|}^{\neg j_{|Y|}} \\
&\succ w \succ \pi_X(|X|) \succ \cdots \succ \pi_X(1) \\
&\succ y_1^{j_1} \succ \cdots \succ y_{|Y|}^{j_{|Y|}} \\
&\succ z_{|Z_0|}^{r_{|Z_0|}} \succ \cdots \succ z_1^{r_1}
\end{aligned}$$

Notice that w is preferred to every alternative in X and is ranked among the upper half of the alternatives by every voter. Thus, w has a strictly better than average score under Borda, Bucklin and Copeland. However, every other alternative $a \in A \setminus \{w\}$ obtains a score that is roughly average, since if voter v_{2i} ranks a in position k, then v_{2i-1} ranks a in position $m - k$. Also notice that each $z \in Z_0$ is placed among the top $|Z_0|$ positions in voter v_{2i} with probability $\frac{1}{2}$, independent of the rankings of the other members of Z_0.

Lemma 1 shows that with high probability, given any N preference profiles $P_1, \cdots, P_N$, it is possible to mix and match voters from the N profiles in such a way that w no longer obtains a good approximation ratio.

Lemma 1. *Let $P_i = (v_1^i, \cdots, v_n^i)$ for $i = 1, \cdots, N$ be random preference profiles drawn from $\mathcal{P}$ and let $\delta \in (0,1)$. There exists a $z \in Z_0$ and a $r \in \{1, \cdots, N\}^n$ such that in $P_r = (v_1^{r_1}, \cdots, v_n^{r_n})$, z is ranked among the top $|Z_0|$ positions by at least $(1-\delta)(1-\frac{1}{2}^{N-1})n$ voters with probability at least $1 - e^{-\frac{|Z_0|(1-\frac{1}{2}^{N-1})n'\delta^2}{2}}$.*

Proof. Recall that for each $i \in \{1, \cdots, n'\}$ and $z \in Z_0$, exactly one of v_{2i}^1 and v_{2i-1}^1 rank z among the top $|Z_0|$ positions. Without loss of generality, assume that v_{2i}^1 ranks z among the top $|Z_0|$ positions.

The probability that for each $j \in \{2, \cdots, N\}$, v_{2i}^j also ranks z among the top $|Z_0|$ positions (rather than v_{2i-1}^j) is $\frac{1}{2}^{N-1}$. Hence, independently for each $i \in \{1, \cdots, n'\}$, with probability $p = 1 - \frac{1}{2}^{N-1}$ there exists indices $j, k \in \{1, \cdots, N\}$ such that v_{2i}^j and v_{2i-1}^k both rank z among the top $|Z_0|$ positions.

By Chernoff bounds, for each $\delta \in (0,1)$, there are fewer than $(1-\delta)pn'$ indices $i \in \{1, \cdots, n'\}$ such that there exists $j, k \in \{1, \cdots, N\}$, where v_{2i}^j and v_{2i-1}^k both rank z among the top $|Z_0|$ positions with probability at most $e^{-\frac{pn'\delta^2}{2}}$.

If at least $(1-\delta)pn'$ such indices i exists, then there exists an $r \in \{1, \cdots, N\}$ such that z is ranked among the top $|Z_0|$ positions by $2(1-\delta)pn' = (1-\delta)pn$ voters, since each such i contributes 2 voters that rank z among the top $|Z_0|$ positions.

As each $z \in Z_0$ is placed independently of the other members of Z_0, the probability that for every $z \in Z_0$ there exist fewer than $(1-\delta)pn'$ indices $i \in \{1, \cdots, n'\}$, such that there exist $j, k \in \{1, \cdots, N\}$ where v_{2i}^j and v_{2i-1}^k both rank z among the top $|Z_0|$ positions, is at most $e^{-\frac{|Z_0|pn'\delta^2}{2}}$. □

Theorem 5 will provide the basis for all of the lower bound proofs.

Theorem 5. *Let $\epsilon \in (0, \frac{1}{2})$, $\delta = 1 - (1-\epsilon)^{\frac{1}{2}}$, $n = 2n'$, and $N = 2 + \lceil \log(\frac{1}{\delta}) \rceil = O(\log(\frac{1}{\epsilon}))$. There exists a set of*

$$2^{\Omega\left(\frac{\epsilon^2}{\log(1/\epsilon)} n|Z_0| - \log(\log(1/\epsilon))\right)}$$

preference profiles S such that

1. *w is ranked in position $|Z_0| + |Y_0| + 1$ by all voters in every preference profile in S,*

2. *For every N preference profiles $P_i = (v_1^i, \cdots, v_n^i)$, $i = 1, \cdots, N$, from S, there exists an $r \in \{1, \cdots, N\}^n$ and $z \in Z_0$, such that in $P_r = (v_1^{r_1}, \cdots, v_n^{r_n})$, z is ranked among the top $|Z_0|$ positions by at least $(1-\epsilon)n$ voters.*

Proof. The proof of Theorem 5 employs the probabilistic method. Let S be a collection of

$$\left[e^{\frac{|Z_0|(1-\frac{\delta}{2})n'\delta^2}{8}}\right]^{\frac{1}{N}} - 1 = 2^{\Omega\left(\frac{\epsilon^2}{\log(1/\epsilon)} n|Z_0|\right)}$$

random preference profiles drawn from $\mathcal{P}$. Note that the collection S may not contain distinct preference profiles, since we sample from $\mathcal{P}$ with replacement.

Clearly, by construction, every preference profile in S satisfies property (1).

Notice that $\frac{1}{2}^{N-1} \leq \frac{\delta}{2}$. Consider N random preference profiles $P_1, \cdots, P_N$ drawn from $\mathcal{P}$. By Lemma 1, the probability that there is no $z \in Z_0$ and $r \in \{1, \cdots, N\}^n$ such that, in P_r there are at least $(1-\frac{\delta}{2})(1-\frac{1}{2}^{N-1})n > (1-\delta)^2 n = (1-\epsilon)n$ voters that rank z among the top $|Z_0|$ positions is

$$e^{-\frac{|Z_0|(1-\frac{1}{2}^{N-1})n'\delta^2}{8}} < e^{-\frac{|Z_0|(1-\frac{\delta}{2})n'\delta^2}{8}}.$$

The probability that the collection S satisfies property (2) is

$$\begin{aligned} Pr[S \text{ satisfies } (2)] &\geq 1 - Pr[S \text{ fails } (2)] \\ &\geq 1 - \binom{|S|}{N} Pr[P_1, \cdots, P_N \text{ fails } (2)] \\ &> 1 - |S|^N Pr[P_1, \cdots, P_N \text{ fails } (2)] \\ &\geq 1 - |S|^N \cdot e^{-\frac{|Z_0|(1-\frac{\delta}{2})n'\delta^2}{8}} \\ &> 0, \end{aligned}$$

where, in the second and third lines, $Pr[P_1, \cdots, P_N \text{ fails } (2)]$ is the probability that N randomly selected preference profiles from $\mathcal{P}$ fails to satisfy property (2).

Since with probability strictly greater than 0, S satisfies property (2), it is concluded that such a collection S exists.

Notice that if a given preference profile P appears in S more than $N-1$ times, then S does not satisfy property (2) (because mixing and matching voters from N copies of P results in another copy of P). Hence, there are necessarily a set of

$$\frac{|S|}{N} = 2^{\Omega\left(\frac{\epsilon^2}{\log(1/\epsilon)} n|Z_0| - \log(\log(1/\epsilon))\right)}$$

distinct preference profiles that satisfies (1) and (2). □

For a given $\epsilon \in (0,1)$, let S_ϵ be the set shown to exist in Theorem 5 and let N_ϵ be the corresponding value of N.

Theorem 5 can be used to prove lower bounds on the communication complexity of approximating Borda, Bucklin, and Copeland.

Theorem 6. *Let $\rho \in (\frac{1}{\sqrt{2}}, 1)$ and $\frac{1}{\sqrt{2}} + \delta = \rho$. The communication complexity of obtaining a rho-approximation to Borda is $\Omega\left(\frac{\delta^3}{\log(1/\delta)} nm - \log(\log(1/\delta))\right)$.*

Proof. Let $c = \frac{1}{2}(1 + \frac{1}{\sqrt{2}\rho})$. Let $\alpha = 1 - \frac{1}{2c\rho}$ and let $\beta = 1 - 2(1-\alpha)^2$. Notice that $\frac{1}{1-\alpha} = 2c\rho$ and $\frac{1-\alpha}{1-\beta} = \frac{1}{2(1-\alpha)}$.

Let m be sufficiently large so that $\left(\frac{m-1}{m-1/(1-\alpha)}\right) < \frac{1}{c}$. Let $\epsilon = 1 - c$. Thus, $\frac{1}{1-\epsilon} = \frac{1}{c}$.

Employing Theorem 5 requires that we must specify how A is partitioned into X, Y, and Z. It suffices to specify the sizes of each set, as we are indifferent to the particular alternatives in each. Let Z_0 and Z_1 each contain βm alternatives and let Y_0 and Y_1 each contain $(\alpha - \beta)m$ alternatives. Thus, $|Z| + |Y| = 2\alpha m$ and $|X| = m - 2\alpha m - 1$. Notice that $\beta < \alpha < \frac{1}{2}$, so A can be partitioned in this manner.

Let S_ϵ be the set shown to exist in Theorem 5. Then

$$\begin{aligned} |S_\epsilon| &= 2^{\Omega\left(\frac{\epsilon^2}{\log(1/\epsilon)} n|Z_0| - \log(\log(1/\epsilon))\right)} \\ &= 2^{\Omega\left(\frac{\delta^3}{\log(1/\delta)} nm - \log(\log(1/\delta))\right)}. \end{aligned}$$

It will be shown that S_ϵ is a ρ-fooling set.

In every preference profile in S_ϵ, w is ranked in position $|Z_0| + |Y_0| + 1 = \alpha m + 1$. Hence, the Borda score of w is $n(m - \alpha m - 1)$ in every preference profile in S. The Borda score of every other alternative is at most $n\frac{m-1}{2}$. Thus, the approximation ratio obtained by any $x \in A \setminus \{w\}$ is

$$\begin{aligned} \frac{sc(x)}{sc(w)} &\leq \frac{n\frac{m-1}{2}}{n((1-\alpha)m - 1)} \\ &= \left(\frac{1}{1-\alpha}\right)\left(\frac{m-1}{m-1/(1-\alpha)}\right)\frac{1}{2} \\ &< (2c\rho) \cdot \frac{1}{c} \cdot \frac{1}{2} \\ &= \rho \end{aligned}$$

Thus, in every preference profile in S, no alternative other than w obtains a *rho*-approximation.

For any N_ϵ preference profiles, there exists a $z \in Z$ and $r \in \{1, \cdots, N_\epsilon\}$ such that in P_r, z is ranked among the top $|Z_0|$ positions by $(1-\epsilon)n$ of the voters. Hence, the Borda score of z is at least $(1-\epsilon)n(m - \beta m)$. In P_r the approximation obtained by w is then

$$\begin{aligned} \frac{sc(w)}{sc(x)} &\leq \frac{n((1-\alpha)m - 1)}{(1-\epsilon)n(1-\beta)m} \\ &< \frac{1-\alpha}{(1-\epsilon)(1-\beta)} \\ &= \frac{1}{2(1-\alpha)(1-\epsilon)} \\ &< \frac{1}{2} \cdot (2c\rho) \cdot \frac{1}{c} \\ &= \rho \end{aligned}$$

Therefore, in any $N_\epsilon = O(\log(1/\delta))$ preference profiles, it is possible to mix voters in such a way that w no longer obtains a *rho*-approximation. Therefore, the communication complexity of computing a *rho*-approximation to Borda is

$$\log\left(\frac{|S_\epsilon|}{N_\epsilon}\right) = \Omega\left(\frac{\delta^3}{\log(1/\delta)} nm - \log(\log(1/\delta))\right).$$

□

Our construction shows that sufficiently good approximations to Borda have communication complexity $\Omega(nm)$. The lower bound for Bucklin is significantly stronger. It is shown that any deterministic communication protocol that computes any constant factor approximation to Bucklin has communication complexity $\Omega(nm)$. Further, non-trivial lower bounds are presented for a number of non-constant approximation ratios.

Theorem 7. *Let $\rho > 1$. The communication complexity of obtaining a ρ-approximation to Bucklin is $\Omega\left(\frac{nm}{\rho^2}\right)$.*

Proof. Let $\alpha = \frac{1}{2(\rho+1)}$ and $\beta = 2\alpha^2$. Let m be sufficiently large so that $\frac{m-1}{m} > \frac{\rho}{\rho+1}$. Let $\epsilon = \frac{1}{4}$.

Let Z_0 and Z_1 each contain βm alternatives and let Y_0 and Y_1 each contain $(\alpha - \beta)m$ alternatives. Thus, $|Z| + |Y| = 2\alpha m$ and $|X| = m - 2\alpha m - 1$. Notice that $\beta < \alpha < \frac{1}{2}$, so A can be partitioned in this manner.

Let S_ϵ be the set shown to exist in Theorem 5, then

$$|S_\epsilon| = 2^{\Omega(n|Z_0|)} = 2^{\Omega\left(\frac{nm}{\rho^2}\right)}.$$

In every preference profile in S_ϵ, w is ranked in position $|Z_0| + |Y_0| + 1 = \alpha m + 1$. Hence, the Bucklin score of w is $\alpha m + 1$. The Bucklin score of $x \in A \setminus \{w\}$ is at least $\frac{m-1}{2}$.

Hence, the approximation obtained by $x \in A \setminus \{w\}$ is

$$\frac{sc(x)}{sc(w)} \geq \frac{\frac{m-1}{2}}{\alpha m} = (\rho+1)\frac{m-1}{m} > \rho.$$

For any N_ϵ preference profiles, there exists a $z \in Z$ and $r \in \{1, \cdots, N_\epsilon\}$ such that in P_r, z is ranked among the top $|Z_0|$ positions by at least $(1-\epsilon)n = \frac{3n}{4}$ of the voters. Hence, the Bucklin score of z is at most βm. In P_r, the approximation obtained by w is then at least

$$\frac{sc(w)}{sc(x)} \geq \frac{\alpha m}{\beta m} = \frac{\alpha}{\beta} = \frac{1}{2\alpha} > \rho.$$

Therefore, in any N_ϵ preference profiles, it is possible to mix voters in such a way that w no longer obtains a ρ-approximation. By Theorem 1, the communication complexity of obtaining a ρ-approximation to Bucklin is

$$\log\left(\frac{|S_\epsilon|}{N_\epsilon}\right) = \Omega\left(\frac{nm}{\rho^2}\right).$$

□

The previous subsection showed that for each $\delta \in (0,1)$, the communication complexity of computing a m^δ approximate winner in Bucklin elections is $O(nm^{(1-\delta)}\log(m))$. The next result provides a lower bound of $\Omega(nm^{(1-2\delta)})$ for all $\delta \in (\frac{1}{2}, 1)$ on the communication complexity of computing a m^δ approximate winner in Bucklin elections.

Theorem 8. *Let $\delta \in (0, \frac{1}{2})$. The communication complexity of obtaining a m^δ approximation to Bucklin is $\Omega(nm^{(1-2\delta)})$.*

Proof Sketch. The Theorem follows from the proof of Theorem 7, by letting $\rho = m^\delta$. The fooling set S_ϵ in the proof of Theorem 7 contains

$$\begin{aligned} |S| &= 2^{\Omega\left(\frac{nm}{\rho^2}\right)} \\ &= 2^{\Omega\left(\frac{nm}{m^{2\delta}}\right)} \\ &= 2^{\Omega\left(nm^{(1-2\delta)}\right)}. \end{aligned}$$

All that needs to be observed is that the partition of A into X, Y, and Z, given the selection of sizes for each in the proof of Theorem 7, is still valid. However, this is easily determined to be true.

Therefore, the communication complexity of computing a m^δ approximation to Bucklin is

$$\log\left(\frac{|S|}{N_\epsilon}\right) = \Omega\left(nm^{(1-2\delta)}\right).$$

□

A lower bound on the communication complexity of computing ρ-approximate winners for Copeland is provided next.

Theorem 9. *Let $\rho \in (\frac{1}{\sqrt{2}}, 1)$ and $\frac{1}{\sqrt{2}} + \delta = \rho$. The communication complexity of obtaining a ρ-approximation to Copeland is $\Omega(\delta nm)$.*

Proof. Let $m = m' + 1$, $c = \frac{1}{2}(1 + \frac{1}{\sqrt{2}\rho})$. Let $\alpha = 1 - \frac{1}{2c\rho}$ and let $\beta = 1 - 2(1-\alpha)^2$. Notice that $\frac{1}{1-\alpha} = 2c\rho$ and $\frac{1-\alpha}{1-\beta} = \frac{1}{2(1-\alpha)}$. Let m' be sufficiently large so that $\frac{1}{(1-\beta)m'} < (1-c)\rho$. Let $\epsilon = \frac{1}{4}$.

Let Z_0 and Z_1 each contain $\beta m'$ alternatives and let Y_0 and Y_1 each contain $(\alpha - \beta)m' - 1$ alternatives. Thus, $|Z| + |Y| = 2\alpha m' - 2$ and $|X| = m' - 2\alpha m' + 2$. Notice that $\beta < \alpha < \frac{1}{2}$, so A can be partitioned in this manner.

Let S_ϵ be the set shown to exist in Theorem 5, then

$$|S_\epsilon| = 2^{\Omega\left(n|Z_0|\right)} = 2^{\Omega\left(\delta nm\right)}.$$

It will be shown that S_ϵ is a ρ-fooling set.

In every preference profile in S_ϵ, w defeats all members of X and ties all members of $Z \cup Y$ in pairwise elections. Hence, the Copeland score of w is $|X| + \frac{|Z|+|Y|}{2} = (1-2\alpha)m' + 2 + \alpha m' - 1 = (1-\alpha)m' + 1$ in every preference profile in S. The Copeland score of every other alternative is at most $\frac{m'}{2}$. Thus, the approximation ratio obtained by any $x \in A \setminus \{w\}$ is

$$\begin{aligned} \frac{sc(x)}{sc(w)} &= \frac{\frac{m'}{2}}{(1-\alpha)m' + 1} \\ &< \frac{1}{2(1-\alpha)} \\ &= c\rho \\ &< \rho. \end{aligned}$$

For any N_ϵ preference profiles, there exists a $z \in Z$ and $r \in \{1, \cdots, N_\epsilon\}$ such that in P_r, z is ranked among the top $|Z_0|$ positions by over half of the voters. Hence, in pairwise elections z defeats all alternatives in X and Y. By construction, if $z_i^0 \in Z$ defeats z in a pairwise election, then z necessarily defeats z_i^1 in a pairwise election. Hence, the Copeland score of z is at least $|X| + \frac{|Z|}{2} + |Y| = (1-2\alpha)m' + 2 + \beta m' + 2(\alpha - \beta)m' - 2 = (1-\beta)m'$.

Likewise, w defeats every alternative in X in a pairwise election. However, if w defeats z_i^0 then z_i^1 necessarily defeats w. Likewise, for the alternatives in Y. Thus, the Copeland score of w is at most $|X| + \frac{|Z|+|Y|}{2} = (1-2\alpha)m' + 2 + \alpha m' - 1 = (1-\alpha)m' + 1$.

Thus, in P_r, the approximation obtained by w is

$$\begin{aligned} \frac{sc(w)}{sc(x)} &\leq \frac{(1-\alpha)m' + 1}{(1-\beta)m'} \\ &= \frac{(1-\alpha)}{(1-\beta)} + \frac{1}{(1-\beta)m'} \\ &= \frac{1}{2(1-\alpha)} + \frac{1}{(1-\beta)m'} \\ &< c\rho + (1-c)\rho \\ &= \rho. \end{aligned}$$

Therefore, the communication complexity of obtaining a *rho*-approximation to Copeland is

$$\log\left(\frac{|S_\epsilon|}{N_\epsilon}\right) = \Omega\left(\delta nm\right).$$

□

4. CONCLUSIONS

This paper presents upper and lower bounds on the communication complexity for computing approximate winners in Borda, Bucklin, and Copeland elections. It is shown that for every $\epsilon > 0$ the communication complexity of computing a $1 - \epsilon$ approximate winner in a Borda election is

$O\left(\log(\frac{1}{\epsilon})nm\right)$. For $\delta \in (0, 1 - \frac{1}{\sqrt{2}})$, we show that computing a $\frac{1}{\sqrt{2}} + \delta$ approximate winner in Borda elections has communication complexity $\Omega\left(\frac{\delta^3}{\log(1/\delta)}nm - \log(\log(1/\delta))\right)$. However, computing the true Borda winner has communication complexity $\Omega(nm\log(m))$.

In Bucklin elections, the communication complexity of computing the true winner is $\Theta(nm)$. We show that for all $\rho > 1$, computing a ρ approximate winner in a Bucklin election has communication complexity $\Omega\left(\frac{nm}{\rho^2}\right)$. Hence, fixed constant factor approximate winners in Bucklin elections cannot be computed with asymptotically less communication than computing the true Bucklin winner. However, we show that for all $\delta \in (0, 1)$, computing a m^δ approximate Bucklin winner has communication complexity $O(nm^{1-\delta}\log(m))$. For $\delta \in (\frac{1}{2}, 1)$, a lower bound on the communication complexity of computing a m^δ approximate Bucklin winner of $\Omega(nm^{(1-2\delta)})$ is presented.

A $\Omega(\delta nm)$ lower bound is also presented for the communication complexity of computing $\frac{1}{\sqrt{2}} + \delta$, $\delta \in (0, 1 - \frac{1}{\sqrt{2}})$ approximate winner in Copeland elections. However, as the communication complexity of determining the true Copeland winner is $\Theta(nm\log(m))$, this lower bound leaves open the possibility of an approximation scheme for Copeland, similar to the scheme presented for Borda.

The lower bounds on the communication complexity for computing approximate Borda and Copeland winners only hold for sufficiently good approximation ratios. It may be the case that worse constant factor approximation ratios can be obtained to these rules with a reduced communication complexity overhead. However, we conjecture that the communication complexity of obtaining any constant factor approximation to Borda and Copeland is $\Omega(nm)$. This is an interesting line of future work.

A second line of future work is the design of non-constant factor approximation protocols similar to the one presented for Bucklin. Along this theme, a construction that allows for non-constant factor lower bound proofs is desirable. The construction presented in this paper is limited to sufficiently small constant factor lower bound proofs for Borda and Copeland. It is also desirable to extend this construction to Maximin elections also.

Finally, a third, and potentially fruitful line of further work is the study of the randomized communication complexity of both exact and approximation winner determination. To the best of our knowledge, the randomized communication complexity of (approximate or exact) winner determination has not be studied.

5. REFERENCES

[1] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, 2009.

[2] Ioannis Caragiannis, Jason A. Covey, Michal Feldman, Christopher M. Homan, Christos Kaklamanis, Nikos Karanikolas, Ariel D. Procaccia, and Jeffrey S. Rosenschein. On the approximability of dodgson and young elections. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 1058–1067, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[3] Ioannis Caragiannis, Christos Kaklamanis, Nikos Karanikolas, and Ariel D. Procaccia. Socially desirable approximations for dodgson's voting rule. In *Proceedings of the 11th ACM conference on Electronic commerce*, EC '10, pages 253–262, New York, NY, USA, 2010. ACM.

[4] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.

[5] Vincent Conitzer and Tuomas Sandholm. Communication complexity of common voting rules. In *Proceedings of the 6th ACM conference on Electronic Commerce*, pages 78–87, 2005.

[6] Meir Kalech, Sarit Kraus, Gal A. Kaminka, and Claudia V. Goldman. Practical voting rules with partial information. *Autonomous Agents and Multi-Agent Systems*, 22:151–182, January 2011.

[7] Eyal Kushilevitz and Noam Nisan. *Communication Complexity.* Cambridge University Press, 1997.

[8] Tyler Lu and Craig Boutilier. Robust approximation and incremental elicitation in voting protocols. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (to appear)*, 2011.

[9] Tyler Lu and Craig Boutilier. Vote elicitation with probabilistic preference models: Empirical estimation and cost tradeoffs. In *Proceedings of the Second Conference on Algorithmic Decision Theory (to appear)*, 2011.

[10] John McCabe-Dansted, Geoffrey Pritchard, and Arkadii Slinko. Approximability of dodgsonÕs rule. *Social Choice and Welfare*, 31:311–330, 2008. 10.1007/s00355-007-0282-8.

[11] Ariel D. Procaccia. Can approximation circumvent gibbard-satterthwaite? In *Proceedings of the 24th Conference on Artificial Intelligence*, pages 836–841, 2010.

[12] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, pages 209–213, 1979.

Session 2D
Social Choice II

Lot-based Voting Rules

Toby Walsh
NICTA and UNSW
Sydney, Australia
toby.walsh@nicta.com.au

Lirong Xia
SEAS, Harvard University
Cambridge, MA 02138, USA
lxia@seas.harvard.edu

ABSTRACT

The Internet Engineering Task Force develops and promotes Internet standards like TCP/IP. The chair of the Task Force is chosen by an election which starts with a set of voters being selected at random from the electorate of volunteers. Selecting decision makers by lottery like this has a long and venerable history, having been used in Athenian democracy over two millennia ago, as well as for over 500 years from the 13th Century to elect the Doge of Venice. In this paper, we consider using such lotteries in multi-agent decision making. We study a family of voting rules called lot-based voting rules. Such rules have two steps: in the first step, k votes are selected by a lottery, then in the second round (the runoff), a voting rule is applied to select the winner based on these k votes. We study some normative properties of such lot-based rules. We also investigate the computational complexity of computing the winner with weighted and unweighted votes, and of computing manipulations. We show that for most lot-based voting rules winner determination and manipulation are computationally hard. Our results suggest that this general technique (using lotteries to selecting some voters randomly) may help to prevent strategic behavior of the voters from a computational point of view.

Categories and Subject Descriptors

J.4 [**Computer Applications**]: Social and Behavioral Sciences–Economics; I.2.11 [**Distributed Artificial Intelligence**]: Multi-agent Systems

General Terms

Algorithms, Economics, Theory

Keywords

social choice, voting, manipulation

1. INTRODUCTION

A central question in computational social choice is whether computational complexity can protect elections from manipulation. For certain voting rules it is NP-hard to compute a beneficial manipulation. Modifications like hybridizing together voting rules have also been proposed to make manipulations NP-hard to compute [8, 12]. Of course, NP-hardness results about the complexity of computing manipulations needs to be treated with caution since NP-hardness is only a worst-case notion and "hard" instances may be rare. See [13, 15] for some recent surveys. Of course, if it is already computationally hard for a manipulator to compute the winner, then intuitively it is likely to be computationally hard also to find a beneficial manipulation. Indeed, there are voting rules like Kemeny's, Dodgson's and Slater's where just computing the winner is NP-hard [4, 1, 2, 9].

In this paper, we show that a simple form of non-determinism also offers a potential escape from manipulation. We study a simple "tweak" to a voting rule that uses a lottery to select a subset of the voters before applying the original voting rule. This tweak is inspired by the election procedure for the Chair of the Internet Engineering Task Force (IETF). The IETF develops and promotes Internet standards like TCP/IP. Every two years, ten people are randomly selected from among the 100 or so eligible volunteers to be the voting members of the nominations committee. This committee then nominates the new Chair using some (unspecified) voting rule.

Similar elections have been used in several other settings. For instance, a complex election procedure involving multiple lotteries was used to select the Doge of Venice for over 500 years. Lotteries were also used in the election of the Archbishop of Novgorod, one of the oldest offices in the Russian Orthodox Church. More recently, a lottery was used in 2004 to select a Citizens' Assembly on Electoral Reform which then voted on changing British Columbia's provincial voting system. In 2006, Ontario ran a similar lottery to decide on its provincial voting system. Several Spanish savings banks use a lottery amongst their account holder to select an assembly that then elects representatives for the account holders. Finally, elections involving lotteries have also been proposed as a means to reform both the British House of Lords, and the US House of Representatives.

It has been suggested that lotteries are more democratic than elections since they are inherently egalitarian and arguably less corruptible [11]. On the other hand, lotteries are not without their issues. For instance, the electorate needs to be confident in the randomness of the selection process. To this end, the IETF has defined a robust, general, public method for making random selections (RFC 3797 - Publicly Verifiable Nominations Committee Random Selection).

Our contributions. We study a family of voting rules, called *lot-based rules*, motivated by the election procedure used to select the Chair of the IETF. Lot-based rules are composed of two steps: in the first step, k votes are selected by a lottery, then in the second step (the runoff), a voting rule (called the *runoff rule*) is applied to select the winner based on these k votes. We study some normative properties of lot-based rules. We investigate the computational complexity of computing the winner of lot-based rules with weighted and unweighted votes, respectively, and of computing a manipulation. We show that for most lot-based voting rules win-

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

ner determination and manipulation are computationally hard. Our results suggest that this general technique (using lotteries to selecting some voters randomly) prevents strategic behavior of the voters from a computational point of view.

Related work. Lot-based rules are a type of randomized voting rule. Gibbard [17] proved that when there are at least 3 candidates, if a randomized voting rule satisfies *Pareto optimality* and a probabilistic version of strategy-proofness, then it must be a probability mixture of dictatorships (called *random dictatorships*). We note that any random dictatorship is a lot-based rule, where $k = 1$, and the runoff rule selects the top-ranked candidate as the winner when there is a single vote.

Conitzer and Sandholm [8] and Elkind and Lipmaa [12] studied another type of hybrid voting systems where manipulations are hard to compute. Their systems have two steps: in the first step, a (possibly randomized) voting rule is used to rule out some candidates, and in the second step another voting rule (not necessarily the same as the one used in the first step) is used to select the winner from the remaining candidates. We note that in the first step of their systems, some *candidates* are eliminated, while in the first step of our lot-based rules, some *voters* are eliminated. In that sense, lot-based rules can also be seen as a universal tweak that adds a pre-round that randomly eliminates some voters, to make voting rules hard to manipulate. It would therefore be interesting to consider even more complex voting systems which do both.

Technically, the winner determination problem studied in this paper is also closely related to the problem of constructive control by adding/deleting votes (CCAV/CCDV) [5]. In the winner determination problem studied in this paper, we are given a lot-based rule, a profile P, a candidate c, and a number $0 \leq p \leq 1$. We are asked to decide whether the probability for c to win is larger than p. In CCAV we are also given a set of new votes P', and we are asked whether c can be made win by adding no more than T votes in P'. In CCDV we are asked whether c can be made win by deleting no more than T votes in P. Suppose for some voting rule, it is NP-hard to compute CCDV where exactly T votes are deleted. Then, winner determination for the corresponding lot-based rule is also NP-hard, where $|P| - T$ votes are randomly selected in the runoff, and we are asked whether the probability for c to win is strictly larger than 0. On the other hand, an algorithm for the counting variant of CCAV (that computes how many ways c can be made win by adding exactly T votes in P') can be used to compute the probability for c to win in P' for the corresponding lot-based rule, where T votes are randomly selected in the runoff.

Finally, we note that for lot-based rules, it is easy for the chair to compute the winner provided computing the winner for the runoff rule is easy. This is different from a voting rule like Kemeny's where computing the winner of a given profile is hard. Whilst computing the winner for lot-based rules is computationally easy, it is nevertheless computationally hard to manipulate such rules. In order for a manipulator to compute the benefits of a false vote, she needs to compute the probability for a given candidate to win, which we will show to be computationally intractable.

2. PRELIMINARIES

Let $\mathcal{C} = \{c_1, \ldots, c_m\}$ be the set of *candidates* (or *alternatives*). A linear order $\succ$ on $\mathcal{C}$ is a transitive, antisymmetric, and total relation on $\mathcal{C}$. The set of all linear orders on $\mathcal{C}$ is denoted by $L(\mathcal{C})$. An n-voter profile P on $\mathcal{C}$ consists of n linear orders on $\mathcal{C}$. That is, $P = (V_1, \ldots, V_n)$, where for every $j \leq n$, $V_j \in L(\mathcal{C})$. The set of all n-profiles is denoted by $\mathcal{F}_n$. We let m denote the number of candidates. A (deterministic) *voting rule* r is a function that maps any profile on $\mathcal{C}$ to a unique winning candidate, that is, $r : \mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots \rightarrow \mathcal{C}$. A *randomized voting rule* is a function that maps any profile on $\mathcal{C}$ to a distribution over $\mathcal{C}$, that is, $r : \mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots \rightarrow \Omega(\mathcal{C})$, where $\Omega(\mathcal{C})$ denotes the set of all probability distributions over $\mathcal{C}$. For any randomized scoring rule r, any profile P, and any alternative c, $(r(P))(c)$ is the probability for c to win.

For any profile P and any pair of candidates $\{c, d\}$, let $D_P(c, d)$ denote the number of times that $c \succ d$ in P minus the number of times that $d \succ c$ in P. The *weighted majority graph (WMG)* is a directed graph whose vertices are the candidates, and there is an edge between every pair of vertices, where the weight on $c \rightarrow d$ is $D_P(c, d)$. We note that in the WMG of any profile, all weights on the edges have the same parity (and whether this is odd or even depends on the number of votes), and $D_P(c, d) = -D_P(d, c)$.

The following are some common voting rules. If not mentioned specifically, ties are broken in the fixed order $c_1 \succ c_2 \succ \cdots \succ c_m$.

• *Positional scoring rules*: Given a *scoring vector* of m integers, $\vec{s}_m = (\vec{s}_m(1), \ldots, \vec{s}_m(m))$, for any vote $V \in L(\mathcal{C})$ and any $c \in \mathcal{C}$, we let $\vec{s}_m(V, c) = \vec{s}_m(j)$, where j is the rank of c in V. For any profile $P = (V_1, \ldots, V_n)$, we let $\vec{s}_m(P, c) = \sum_{j=1}^{n} \vec{s}_m(V_j, c)$. The rule selects $c \in \mathcal{C}$ so that $\vec{s}_m(P, c)$ is maximized. We assume scores are decreasing. Examples of positional scoring rules are *plurality*, for which the scoring vector is $(1, 0, \ldots, 0)$, *majority* which is the special case of plurality in which $m = 2$, and *Borda*, for which the scoring vector is $(m-1, m-2, \ldots, 0)$.

• *STV*: This rule requires up to $m - 1$ rounds. In each round, the candidate with the least number of voters ranking them first is eliminated until one of the remaining candidates has a majority.

• *Approval*: Each voter submits a set of candidates (that is, the candidates that are "approved" by the voter). The winner is the candidate approved by the largest number of voters. Every voter can approve any number of candidates.

• *Voting trees*: A voting tree is a binary tree with m leaves, where each leaf is labelled with a candidate. Each internal node is labelled with the child candidate that wins a pairwise election. The candidate labelling the root of the tree (i.e. wins all its rounds) is the winner. The rule that uses a balanced voting tree is the *Cup* rule.

• *Copeland*: We compare every pair of candidates. Each candidate gets 1 point every time it is preferred by more than half the voters. The candidate with the highest total score wins.

• *Maximin*: A candidate's score in a pairwise election is the number of voters that prefer it over the opponent. A candidate's overall score is the lowest score it gets in any pairwise election. The candidate with the highest overall score wins.

• *Ranked pairs*: We consider every pair of candidates in turn, starting with the pair not yet considered in which there is the greatest majority of voters who prefer the first candidate to the second. We construct a ranking which fixes the first candidate above the second unless, by transitivity, this contradicts a previous decision. The candidate at the top of this ranking wins.

In this paper, when we define a linear order, we sometimes do not explicitly specify the rankings among a set of candidates. In such cases, the candidates are ranked according to ascending order of their subscripts. For example, let $m = 4$, $[c_2 \succ \text{Others}]$ represents the linear order $[c_2 \succ c_1 \succ c_3 \succ c_4]$. For any set of candidates C, $\text{Rev}(C)$ represents the linear order where candidates in C are ranked according to the descending order of their subscripts. For example, $[c_2 \succ \text{Rev(Others)}]$ represents the linear order $[c_2 \succ c_4 \succ c_3 \succ c_1]$.

3. LOT-BASED VOTING RULES

We define lot-based voting rules as follows.

DEFINITION 1. *Let X denote a voting rule (deterministic or* *randomized). We define a randomized voting rule LotThenX as follows. Let k be a fixed number that is no more than the number of voters. The winner is selected in two steps: in the first step, k voters are selected uniformly at random, then, in the second step, the winner is chosen by applying the voting rule X to the votes of the k voters selected in the first step.*

For instance, LotThenApproval is an instance of this rule in which the set of voters is first reduced by a lottery, and then a winner is chosen by approval voting. All lot-based rules are parameterized by k, which is the number of randomly selected runoff voters. We emphasize that in the first step of lot-based rules, some *voters* are eliminated, while in the first step of voting systems studied by Conitzer and Sandholm [8] and Elkind and Lipmaa [12], some *candidates* are eliminated.

We first consider the axiomatic properties possessed by lot-based voting rules. As the rules are non-deterministic, we need probabilistic versions of the usual axiomatic properties.[1]

DEFINITION 2. *A randomized voting rule r satisfies*

• anonymity, *if for any profile $P = (V_1, \dots, V_n)$, any permutation π over $\{1, \dots, n\}$, and any candidate c, we have $r(P)(c) = r(V_{\pi(1)}, \dots, V_{\pi(n)})(c)$, where $r(P)(c)$ is the probability of c in the distribution $r(P)$;*

• neutrality, *if for any profile P, any permutation M over $\mathcal{C}$, and any candidates c, we have $r(P)(c) = r(M(P))(M(c))$;*

• unanimity, *if for any profile P where all voters rank c in their top positions, we have $r(P)(c) = 1$;*

• weak monotonicity, *if for any candidate c and any pair of profiles P and P', where P' is obtained from P by raising c in some votes without changing the orders of the other candidates, we have $r(P)(c) \leq r(P')(c)$;*

• strong monotonicity *(a.k.a.* Maskin monotonocity*), if for any candidate c and any pair of profiles $P = (V_1, \dots, V_n)$ and $P' = (V'_1, \dots, V'_n)$, such that for every $j \leq n$ and every $d \in \mathcal{C}$, $c \succ_{V_j} d \Rightarrow c \succ_{V'_j} d$, we have $r(P)(c) \leq r(P')(c)$;*

• Condorcet consistency, *if whenever there exists a candidate who beats all the other candidates in their pairwise elections, this candidate wins the election with probability* 1.

When the voting rule is deterministic (i.e. the unique winner wins with probability 1), all these properties reduce to their counterparts for deterministic rules. The next two theorems show that LotThen$X$ preserves some (but not all) of the axiomatic properties of X.

THEOREM 1. *If the voting rule X satisfies anonymity/ neutrality/ (strong or weak) monotonicity/ unanimity, then for every k, LotThenX also satisfies anonymity/ neutrality/ (strong or weak) monotonicity/ unanimity.*

The proofs are quite straightforward, and are omitted due to space constraints. However, there are other properties that can be lost like, for instance, Condorcet consistency.

THEOREM 2. *LotThenX may not be Condorcet consistent even when X is.*

Proof: Suppose $n = 2k + 1$, $k + 1$ voters vote in one way and the remaining k voters vote in the reverse order. The lottery may select only the votes of the minority, which means that the Condorcet winner loses. □

We note that when $n = k$, LotThen$X$ becomes exactly X. Therefore, if X does not satisfy some axiomatic property, neither does LotThen$X$.

[1]Definitions of the axiomatic properties for approval are omitted due to the space constraints.

THEOREM 3. *If LotThenX satisfies an axiomatic property for every k, then X also satisfies the same axiomatic property.*

4. COMPUTING THE WINNER

In the remainder of this paper, we focus on the case where $k < n$, that is, when lot-based voting rules are non-deterministic. Hence, even if we know all the votes, we can only give a probability in general that a certain candidate wins. The EVALUATION problem we study is defined similar to the evaluation problem defined in [10].

DEFINITION 3. *In an* EVALUATION *problem, we are given a lot-based rule r, a profile P, a number p in $[0, 1]$, and a candidate c. We are asked to compute whether $(r(P))(c) > p$.*

We note that in EVALUATION, the number of runoff voters k is a part of the input. In this section, we show that lot-based voting rules may provide some resistance to strategic behavior by making it computationally hard even to evaluate who may have won. In particular, we show that there exist deterministic voting rules for which computing the winner is in P, but EVALUATION of the corresponding lot-based voting rule is NP-hard. As is common in computational social choice, we consider both weighted voted with a small number of candidates, and unweighted votes with an unbounded number of candidates. Of course, even if EVALUATION is hard, the manipulator may still be able to compute an optimal strategy in polynomial time. This issue will be discussed in Section 5.

4.1 Weighted votes

With weighted votes, computing who wins the Cup or Approval rule is polynomial. On the other hand, deciding if a candidate wins LotThenCup or LotThenApproval with greater than some probability is computationally intractable.

THEOREM 4. EVALUATION *for LotThenCup is* NP*-hard when votes are weighted and there are three or more candidates.*

Proof: We give a reduction from a special SUBSET-SUM. In such a SUBSET-SUM problem, we are given $2k'$ integers $\mathcal{S} = \{w_1, \dots, w_{2k'}\}$ and another integer W. We are asked whether there exists $S \subset \mathcal{S}$ such that $|S| = k'$ and the integers in S sum up to W. We consider the cup rule (balanced voting tree) where ties are broken in lexicographical order. We only show the proof for three candidates; other cases can be proved similarly. For any SUBSET-SUM instance, we construct an EVALUATION for LotThenCup instance as follows.
Candidates: $\mathcal{C} = \{a, b, c\}$. The cup rule has a play b and the winner of this play c. Let $k = k' + 1$.
Profile: For each $i \leq 2k'$, we have a vote $c \succ a \succ b$ of weight w_i. In addition, we have one vote $b \succ a \succ c$ of weight W. We consider the problem of evaluating whether candidate a can win with some probability strictly greater than zero.

If the lottery does not pick $b \succ a \succ c$, then c wins for sure. If the lottery picks the vote $b \succ a \succ c$, then there are three cases to consider. In the first case, the sum of the weights of the other k' votes is strictly less than W. Then, b beats a in the first round, so a does not win. In the second case, the sum of the weights of the other k' votes is strictly more than W. Then, a beats b in the first round, but then loses to c in the second round, so a does not win. In the third case, the sum of weights of the other k' votes is exactly W. Then, a wins both rounds due to tie-breaking. Hence a wins if and only if the sum of the weights of the remaining k' votes is exactly W. Thus the probability that a wins is greater than zero if and only if there is a subset of k' integers with sum W. □

THEOREM 5. *There is a polynomial-time Turing reduction from* SUBSET-SUM *to* EVALUATION *for LotThenApproval with weighted votes and two candidates.*[2]

Proof sketch: Given any SUBSET-SUM instance $\{w_1, \ldots, w_{2k'}\}$ and W, we construct the following two types of EVALUATION for LotThenApproval instances: the profiles in both of them are the same, but the tie-breaking mechanisms are different. For each $i \leq 2k'$, there is a voter with weight w_i who approves candidate a. In addition, there is voter with weight W who approves b. Let P denote the profile and $k = k' + 1$. For any $p \in [0, 1]$, we let $A(p)$ (respectively, $B(p)$) denote the EVALUATION instance where ties are broken in favor of a (respectively, b), and we are asked whether the probability that a (respectively, b) wins for P is strictly larger than p. Then, we use binary search to search for an integer i such that $i \in [0, \binom{2k'+1}{k'} - \binom{2k'}{k'+1}]$ and the answers to both $A\left(1 - \frac{i+1}{\binom{2k'+1}{k'+1}}\right)$ and $B\left(\frac{i}{\binom{2k'+1}{k'+1}}\right)$ are "yes". If such an i can be found, then the SUBSET-SUM instance is a "yes" instance; otherwise it is a "no" instance. □

It follows that, with weighted votes and two candidates, if EVALUATION for LotThenApproval is in P then P=NP.

4.2 Unweighted votes

When the number of candidates is bounded above by a constant, computing the probability for a candidate to win for LotThen$X$ is in P for any anonymous voting rule X. Algorithm 1 uses dynamic programming, and exploits the fact that when the number of candidates is bounded above by a constant, the number of different profiles of n votes for anonymous voting rules is polynomial in n (no more than $n^{m!}$). For anonymous rules, it suffices to characterize a profile by an $m!$ dimensional vector (called a *voting situation*), where each dimension corresponds to a linear order V, and the component represents how many copies of V in the profile. For each natural number t, we let $D_t \subseteq \mathbb{N}^{m!}_{\geq 0}$ denote the set of all vectors whose components sum up to t.

Algorithm 1: Evaluation

```
  Input: LotThenX, a profile $P \in D_n$, a candidate $c$.
  Output: The probability $p(P)$ for $c$ to win.
1 for each $P_t \in D_t$ do
2     Let $p(P - P_t) = \begin{cases} 1 & \text{if } P - P_t \geq \vec{0} \text{ and } X(P_t) = c \\ 0 & \text{otherwise} \end{cases}$,
      where $(P - P_t)$ is a vector in $\mathbb{N}^{m!}_{\geq 0}$ ;
3 end
4 for $l = t - 1$ to 0 do
5     for each $P_l \in D_l$ do
6         Let $p(P - P_l) = \sum_{\vec{e} \in D_1} \frac{1}{m!} \cdot p(P - P_l - \vec{e})$;
7     end
8 end
9 return $p(P)$;
```

Even though Algorithm 1 runs in polynomial time, its complexity is still very high in the worst case. For example, when $m = 5$, Algorithm 1 runs in time $\Theta(n^{120})$. We next show that when the number of candidates is unbounded, EVALUATION for LotThen$X$ is hard to compute for many common voting rules including Borda, Copeland, Maximin and Ranked Pairs.

[2]The proof can be easily extended to any LotThen$X$ where X is the same as the majority rule when there are only two candidates.

THEOREM 6. *With unweighted votes and an unbounded number of candidates,* EVALUATION *for LotThenBorda is* NP*-hard.*

Proof: We prove the NP-hardness by a reduction from the EXACT 3-COVER (X3C) problem [16]. In an X3C instance, we are given a set $\mathcal{V} = \{v_1, \ldots, v_{3q}\}$ of $3q$ elements and $\mathcal{S} = \{S_1, \ldots, S_t\}$ such that for every $i \leq t$, $S_i \subseteq \mathcal{V}$ and $|S_i| = 3$. We are asked whether there exists a subset $J \subseteq \{1, \ldots, t\}$ such that $|J| = q$ and $\bigcup_{j \in J} S_j = \mathcal{V}$.

For any X3C instance $\mathcal{V} = \{v_1, \ldots, v_{3q}\}$ and $\mathcal{S} = \{S_1, \ldots, S_t\}$, we construct an EVALUATION instance for LotThenBorda as follows.

Candidates: $\mathcal{C} = \{c\} \cup \mathcal{V} \cup D$, where $D = \{d_1, \ldots, d_{3q^2}\}$. Let $k = q$.

Profile: For each $j \leq t$, we let $V_j = [(\mathcal{S} \setminus S_j) \succ c \succ D \succ S_j]$. The profile is $P = (V_1, \ldots, V_t)$. We are asked to compute whether the probability for c to win is larger than zero ($p = 0$).

Suppose the EVALUATION instance has a solution. Then, there exists a sub-profile P' of P such that $|P'| = q$ and Borda$(P') = c$. Let $P' = (V_{i_1}, \ldots, V_{i_q})$. We claim that $J = \{i_1, \ldots, i_q\}$ constitutes a solution to the X3C instance. Suppose there exists a candidate $v \in \mathcal{V}$ that is not covered by any S_j where $j \in J$. Then, v is ranked above c in each vote in P', which contradicts the assumption that c is the Borda winner.

Conversely, let $J = \{i_1, \ldots, i_q\}$ be a solution to the X3C instance. Let $P' = (V_{i_1}, \ldots, V_{i_q})$. It follows that for each $v \in \mathcal{V}$, the Borda score of c minus the Borda score of v is at least $3q^2 - (3q - 3) \times q > 0$. For each $d \in D$, c is ranked above d in each vote in P'. Therefore, c is the Borda winner, which means that the EVALUATION instance is an "yes" instance. □

THEOREM 7. *With unweighted votes and an unbounded number of candidates, computing the probability for a given candidate to win under LotThenBorda is* #P*-complete.*

Proof: We prove the theorem by a reduction from the #PERFECT-MATCHING problem. Given three sets $X = \{x_1, \ldots, x_t\}$, $Y = \{y_1, \ldots, y_t\}$, and $E \subseteq X \times Y$, a *perfect matching* is a set $J \subseteq E$ such that $|J| = t$, and all elements in X and Y are covered by J. In a #PERFECT-MATCHING instance, we are asked to compute the number of perfect matchings. Given any #PERFECT-MATCHING instance X, Y, and E, we construct the following instance of computing the winning probability of a given candidate for LotThenBorda.

Candidates: $\mathcal{C} = \{c, b\} \cup X \cup Y \cup A$, where $A = \{a_1, \ldots, a_{2t}\}$. Let $k = 2t$. Suppose ties are broken in the following order: $X \succ Y \succ c \succ$ Others. We are asked to compute the probability that c wins.

Profile: For each edge $(x_i, y_j) \in E$, we first define a vote $W_{i,j} = [X \succ a_i \succ c \succ Y \succ b \succ \text{Others}]$, where elements within Y, X, A_i and B_j are ranked in ascending order of their subscripts. Then, we obtain $V_{i,j}$ from $W_{i,j}$ by exchanging the positions of the following two pairs of candidates: (1) x_i and a_i; (2) y_j and b. Let $P_V = \{V_{i,j} : \forall (x_i, y_j) \in E\}$.

For each $j \leq t$, we define a vote $U_j = [\text{Rev}(Y) \succ c \succ a_{t+j} \succ \text{Rev}(X) \succ \text{Others}]$, where Rev$(X)$ is the linear order where the candidates in X are ranked in descending order of their subscripts. Let $P_U = \{U_1, \ldots, U_t\}$. Let the profile be $P = P_V \cup P_U$.

Let P' be a sub-profile of P such that $|P'| = k = 2t$. We first claim that if Borda$(P') = c$, then $P_U \subseteq P'$. For the sake of contradiction, suppose $P_V \cap P' = \{V_{i_1,j_1}, \ldots, V_{i_l,j_l}\}$, where $l > t$. Because $|X| = t$, there exists $i \leq t$ such that i is included in the multiset $\{i_1, \ldots, i_l\}$ at least two times. For any candidate c', let $s(P, c')$ denote the Borda score of c' in P. It follows that $s(P, x_i) > s(P, c)$, which contradicts the assumption that c is the Borda winner.

Next, we prove that for any $P' = P_U \cup \{V_{i_1,j_1}, \dots, V_{i_t,j_t}\}$ such that Borda$(P') = c$, $J = \{(x_{i_1}, y_{j_1}), \dots, (x_{i_t}, y_{j_t})\}$ is a perfect matching. Suppose J is not a perfect matching. If $x \in X$ (respectively, $y \in Y$) is not covered by J, then we have $s(P, x) = s(P, c)$ (respectively, $s(P, y) = s(P, c)$), which means that c is not the Borda winner due to tie-breaking. This contradicts the assumption. We note that different P' correspond to different perfect matchings. Similarly, any perfect matching corresponds to a different profile P' such that $|P'| = 2t$ and Borda$(P') = c$. We note that the probability that c wins is the number of such P' divided by $\binom{t+|E|}{2t}$. Therefore, computing the probability for c to win is #P-hard. It is easy to check that computing the probability for c to win is in #P. □

It has been shown that computing constructive control by deleting votes is NP-complete [14]. Therefore, we have the following corollary.

COROLLARY 1. *With unweighted votes and an unbounded number of candidates,* EVALUATION *for LotThenCopeland and for LotThenMaximin is* NP*-hard.*

THEOREM 8. *With unweighted votes and an unbounded number of candidates,* EVALUATION *for LotThenRankedPairs is* NP*-hard.*

Proof: We prove the NP-hardness by a reduction from a special X3C problem $\mathcal{V} = \{v_1, \dots, v_{3q}\}$ and $\mathcal{S} = \{S_1, \dots, S_t\}$, where $t \geq 3q$ and t is even. (If $t < 3q$ then we add $3q - t$ copies of S_1 to $\mathcal{S}$, and if t is odd then we add 1 copy of S_1 to $\mathcal{S}$.) For any element v_i, we let $\Delta(v_i)$ denote the number of times c_i is covered by S_j. For any X3X instance where $t \geq 3q$ and t is even, we construct the following EVALUATION instance.

Candidates: $\mathcal{V} \cup \{c, d, e\}$. Ties are broken in the order $d \succ e \succ c \succ \mathcal{V}$. Let $k = |P| - q$.

Profile: Let P denote a profile composed of the votes shown in Table 1. We are asked to compute whether the probability for c to win is larger than zero ($p = 0$).

#	Votes
P_1: for each $j \leq t$	$d \succ e \succ S_j \succ c \succ (\mathcal{V} \setminus S_j)$
P_2: $t/2 - q + 1$	$c \succ d \succ e \succ \mathcal{V}$ Rev$(\mathcal{V}) \succ c \succ e \succ d$
P_3: $q - 1$	$c \succ d \succ e \succ \mathcal{V}$ Rev$(\mathcal{V}) \succ e \succ c \succ d$
P_4: for each $i \leq 3q$, $t/2 - \Delta(v_i)$	$d \succ v_i \succ c \succ e \succ$ Others Rev(Others) $\succ v_i \succ e \succ c \succ d$
P_5: for each $i \leq 3q$, $t/2 - \Delta(v_i)$	$d \succ c \succ e \succ v_i \succ$ Others Rev(Others) $\succ e \succ v_i \succ c \succ d$

Table 1: The profile *P* for LotThenRankedPairs.

In the profile, P_1 is used to encode the X3C instance; P_2 and P_3 are used to reduce the weights on the edge $d \to c$ and $e \to c$ in the weighted majority graph; P_4 is used to reduce the weights on the edges $v_i \to c$, and P_5 is used to balance the weight loss on $e \to v_i$ introduced in P_4. We make the following observation on the weighted majority graph of P.

• There is an edge $d \to e$ with weight t, an edge $e \to c$ with weight $2q - 2$. The edge between c and d has zero weight.

• For any $i \leq 3q$, there is an edge $d \to v_i$ with weight t, an edge $e \to v_i$ with weight t, and all edges between c and v_i have zero weight.

Suppose we remove q votes from P, then because $t \geq 3q$, we have that $d \to e$, $d \to v_i$ and $e \to v_i$ are fixed in the final order. We note that $\{d, e\} \succ c$ only in votes in P_1 . Therefore, if q votes can be eliminated to make c win for ranked pairs, then in all of them, we must have $\{d, e\} \succ c$, otherwise $d \to e$ and $e \to c$ will be fixed before $c \to d$ is considered. It follows that the q eliminated votes must come from P_1. Moreover, in order for c to win, the weight on each edge from $\mathcal{V}$ to c should be no more than $q - 2$, otherwise a path from d via some candidates in $\mathcal{V}$ will be fixed before $c \to d$ is considered. This means that the eliminated q votes in P_1 correspond to an exact cover of $\{v_1, \dots, v_{3q}\}$. Therefore, EVALUATION for LotThenRankedPairs is NP-hard. □

For LotThenPlurality, we have the following corollary, which follows from a polynomial-time dynamic programming algorithm that solves the counting variant of CCAV in [22].

COROLLARY 2. *With unweighted votes and an unbounded number of candidates, computing the probability for a given candidate to win under LotThenPlurality can be solved in polynomial time.*

5. MANIPULATION

Suppose there are a group of manipulators who know the vote of the non-manipulators. We consider the computational complexity for the manipulators to compute (perhaps non-truthful) votes so that a preferred candidates wins the election. We limit our attention to unweighted votes. We consider two types of manipulation problem defined as follows.

DEFINITION 4. *In a* fixed manipulation *problem, given the votes of the non-manipulators, a favoured candidate and a probability p, we ask if the manipulator(s) can cast fixed vote(s) so that the candidate wins with probability greater than p. In an* improving manipulation *problem, we are not given any p but are given the truthful vote of the manipulator(s) and we ask if the manipulator(s) can cast fixed vote(s) so that the probability of the given candidate winning increases.*

An interesting extension, which we leave for future work, is when the manipulator(s) can decide how to vote after the lottery has taken place. This will increase the opportunities for manipulation.

It is easy to see that the improving manipulation problem for LotThenPlurality can be computed in polynomial time: the optimal strategy for the manipulator(s) is to vote for c. Therefore, it follows from Corollary 2 that fixed manipulation for LotThenPlurality is in P. By Algorithm 1, when the number of candidates and the number of manipulators are bounded above and the voting rule X is anonymous, both the fixed and the improving manipulation problems are in P.

COROLLARY 3. *When the number of candidates and the number of manipulators are bounded and votes are unweighted, fixed or improving manipulation of LotThen$X$ is in* P *for any anonymous rule X.*

When the number of candidates is not bounded, adding a lottery can increase the complexity of computing a manipulation. For example, when there is only one manipulator, computing a manipulation for Borda is in P, but it is NP-hard to compute both fixed and improving manipulations of LotThenBorda.

THEOREM 9. *When the number of candidates is unbounded and votes are unweighted, fixed manipulation of LotThenBorda is* NP*-hard for even a single manipulator.*

Proof: We prove the NP-hardness by a reduction from X3C that is similar to the reduction in the proof of Theorem 6. Given an X3C instance $\mathcal{V} = \{v_1, \dots, v_{3q}\}$ and $\mathcal{S} = \{S_1, \dots, S_t\}$, we construct the following manipulation instance for LotThenBorda as follows.

Candidates: $\mathcal{C} = \{c\} \cup \mathcal{V} \cup D$, where $D = \{d_1, \dots, d_{3q^2}\}$. Let $k = q$.
Profile: For each $j \leq t$, we let $V_j = [(\mathcal{S} \setminus S_j) \succ c \succ D \succ S_j]$. The profile is $P = (V_1, \dots, V_t)$. Let $p = \binom{t-1}{q-1}/\binom{t}{q}$.

We claim that in this instance the optimal strategy for the manipulator is to vote for $[c \succ D \succ \mathcal{S}]$. We note that if the manipulator is not eliminated by the lottery, then c must win. This happens with probability $\binom{t-1}{q-1}/\binom{t}{q}$. If the manipulator is eliminated by the lottery, then following the same reasoning in the proof of Theorem 6, the probability for c to wins is strictly larger than 0 if and only if the X3C instance has a solution. This proves that fixed manipulation of LotThenBorda is NP-hard. □

THEOREM 10. *When the number of candidates is unbounded and votes are unweighted, improving manipulation of LotThenBorda is* NP*-hard for even a single manipulator.*

Proof: We prove the NP-hardness by a reduction from X3C. Given an X3C instance $\mathcal{V} = \{v_1, \dots, v_{3q}\}$ and $\mathcal{S} = \{S_1, \dots, S_t\}$, we construct the following manipulation instance for LotThenBorda as follows. W.l.o.g. q is even (otherwise we add three new elements $\{v_{3q+1}, v_{3q+2}, v_{3q+3}\}$ to $\mathcal{V}$ and $\{v_{3q+1}, v_{3q+2}, v_{3q+3}\}$ to $\mathcal{S}$).
Candidates: $\mathcal{C} = \{c\} \cup \mathcal{V} \cup D \cup E$, where $D = \{d_1, \dots, d_{3q^2}\}$, $E = \{e_1, \dots, e_{3q^2}\}$. Let $k = 3q/2$.
Profile: We will construct the profile in the way such that (1) if the manipulator vote for $[c \succ D \succ E \succ \mathcal{S}]$ and is not eliminated by the lottery, then the probability for c to win is non-zero if and only if the X3C instance has a solution, and (2) if the manipulator vote for $[\mathcal{S} \succ D \succ E \succ c]$ and is not eliminated by the lottery, then c never wins. Therefore, even though $[c \succ D \succ E \succ \mathcal{S}]$ seems better than $[\mathcal{S} \succ D \succ E \succ c]$, it is hard for the manipulator to figure out whether the former is strictly better. More precisely, for each $j \leq t$, we let

$$V_j = [(\mathcal{S} \setminus S_j) \succ D \succ c \succ E \succ S_j]$$

$$\text{and } U_j = [(\mathcal{S} \setminus S_j) \succ \text{Rev}(E) \succ c \succ \text{Rev}(D) \succ S_j]$$

Let $P_1 = \{V_1, \dots, V_t\}$ and $P_2 = \{U_1, \dots, U_t\}$. Let $E' = \{e_1, \dots, e_{3q+20}\}$. Let P_3 consist of $q/2 - 1$ copies of

$$[c \succ D \succ (E \setminus E') \succ \mathcal{S} \succ E']$$

The profile is $P = P_1 \cup P_2 \cup P_3$. We are asked whether the manipulator can find a vote better than $W = [\mathcal{S} \succ D \succ E \succ c]$.

Suppose the X3C instance has a solution, w.l.o.g. denoted by $\{S_1, \dots, S_q\}$. We prove that if the manipulator votes for $W' = [c \succ D \succ E \succ \mathcal{S}]$, then the probability for c to win is higher than in the case where she votes for W. We note that if the lottery eliminates the manipulator, the probability for c to win cancels out. Therefore, we only need to focus on the lotteries where the manipulator is selected. We note that if the manipulator votes for V, then c cannot win. If the manipulator votes for W' and the lottery chooses her and $\{V_1, \dots, V_{q/2}, U_{q/2+1}, \dots, U_q\} \cup P_3$, then c is the Borda winner, which means that there is an improving manipulation.

On the other hand, suppose that there is an improving manipulation. We claim that if the lottery selects the manipulator and c is the Borda winner, then (1) all votes in P_3 must be selected, and (2) the votes selected in $P_1 \cup P_2$ constitute an exact cover. If (1) or (2) is not satisfied, then in the profile after the lottery without the manipulator's vote, there exists a candidate in $\mathcal{V}$ whose Borda score is higher than the Borda score of c by at least $2|D| + 3q$, which contracts the assumption that c is the Borda winner when we also take into account the manipulator's vote. Therefore, the X3C instance has a solution. □

LotThen$X$ often inherits any computational resistance to manipulation that the voting rule X may have. For example, LotThenSTV inherits the computational complexity of STV against manipulation [3].

THEOREM 11. *With unweighted votes and an unbounded number of candidates, for even a single manipulator, fixed and improving manipulation are* NP*-hard for LotThenSTV.*

Proof: We prove the NP-hardness by a reduction from a special unweighted coalitional manipulation problem for STV with one manipulator (UCM$_1$) where c is ranked in the top position in at least one vote in P^{NM}. This problem is NP-complete [3]. For any UCM$_1$ instance (STV, P^{NM}, c) where c is ranked in the top position in at least one vote in P^{NM} ($|P^{NM}| = n - 1$), we construct the following manipulation problem. Let $\mathcal{C}'$ denote the set of candidates in the UCM$_1$ instance.

Candidates: $\mathcal{C}' \cup \{d\}$, where d is an auxiliary candidate.

Profile: Let P denote a profile of $2n - 1$ votes as follows. The first $n-1$ votes, denoted by P_1, are obtained from P^{NM} by putting d right below c. The next n votes, denoted by P_2, all rank d in the first position (other candidates are ranked arbitrarily). Let $k = |P| - 1$ and $p = 0$. For the improving manipulation problem, we let W be an arbitrary vote where d is ranked in the top position.

We note that if none of votes in P_2 is eliminated, then d is the winner, because it is already ranked in the top position by more than half the votes. Therefore, the only way c can win is if some voter in P_2 is eliminated in the first round.

Suppose the UCM$_1$ instance has a solution, denoted by V. Then, let V' denote the linear order over $\mathcal{C}' \cup \{d\}$ obtained from V by ranking d in the bottom position. Let P' denote the profile where a vote in P_2 is eliminated by the lot. We note that d is ranked in the top position $n - 1$ times in P'. Therefore, d is never eliminated in the first $|\mathcal{C}'| - 1$ rounds. Moreover, for any $j \leq |\mathcal{C}'| - 1$, the candidate that is eliminated in the jth round for P' is exactly the same as the candidate that is eliminated in the jth round for $P^{NM} \cup \{V\}$. In the last round, c is ranked in the top position n times, which means that STV$(P') = c$. Hence, the probability c wins is strictly larger than 0, which is the probability c wins if the manipulator votes for W.

On the other hand, suppose the manipulator can cast a vote V' to make c win with a non-zero probability. As we have shown above, c wins only when a voter in P_2 is eliminated in the first round. Let $P' = (P_{-n}, V')$. We note in STV for P', d must be eliminated in the last round, because d is ranked in the top position at least $n - 1$ times. Moreover, we recall that c is ranked in the first position in at least one vote in P^{NM}, and d is ranked right below c in the corresponding vote in P'. Therefore, d beats all candidates in $\mathcal{C}' \setminus \{c\}$ in their pairwise elections, which means that in the last round the only remaining candidates must be c and d. Let V be a linear order obtained from $V'_{n'}$ by removing d. It follows that V is a solution to the UCM$_1$ instance.

Therefore, it is NP-hard to compute a fixed or improving manipulation for LotThenSTV, even with a single manipulator. □

Similarly LotThenRankedPairs inherits the computational complexity of RankedPairs against manipulation [23].

THEOREM 12. *With unweighted votes and an unbounded number of candidates, for even a single manipulator, fixed and improving manipulation are* NP*-hard for LotThenRankedPairs.*

Proof: We prove the NP-hardness by a reduction from a special UCM$_1$ problem for ranked pairs, where n is odd ($|P^{NM}| = n-1$), and no weight in the majority graph is larger than $n - 5$ (if there is, then we tweak the instance by adding two pairs of votes $\{[c \succ c_1 \succ$

$\cdots \succ c_{m-1}], [c_{m-1} \succ c_{m-2} \succ \cdots \succ c_1 \succ c]\})$. This problem is NP-complete [23]. For any such UCM$_1$ instance (RP, P^{NM}, c) where n is odd, we construct the following manipulation problem. Let $\mathcal{C}' = \{c, c_1, \ldots, c_{n-1}\}$ denote the set of candidates in the UCM$_1$ instance.

Candidates: $\mathcal{C}' \cup \{d, e\}$, where d and e are auxiliary candidates.

Profile: Let P denote a profile of $3n-2$ votes as follows.

• The first $n-1$ votes are obtained from P^{NM} by putting $d \succ e$ right below c.

• The remaining votes are defined in the following table.

#	Votes
n	$d \succ e \succ$ Others $\succ c$
$(n-1)/2$	$d \succ c \succ e \succ$ Rev(Others)
$(n-1)/2$	$e \succ c \succ d \succ$ Rev(Others)

Let $k = |P| - 1$ and $p = 0$. For the improving manipulation problem, let $W = [d \succ e \succ \text{Others} \succ c]$.

Let P' denote the profile obtained from P by removing one vote of $[d \succ e \succ \text{Others} \succ c]$. We make the following observation on the weighted majority of P'.

• The sub-graph for candidates in $\mathcal{C}'$ is the same as the weighted majority graph of the UCM$_1$ instance.

• There is an edge from d to e with weight $2(n-1)$.

• There are no edges between d and c, and e and c.

• The weights on the edges from d or e to $\mathcal{C}' \setminus \{c\}$ is $n-1$.

Therefore, in the final ranking, it is fixed that $d \succ e \succ (\mathcal{C}' \setminus \{c\})$. Suppose ties among edges are broken in the order where $e \to c$ is fixed before $c \to d$, whenever there is a tie (or alternatively, we can first obtain a set of candidates who win with some ways to break ties among edges, and then use the fixed tie-breaking $d \succ e \succ c \succ$ Others to select the winner). We note that if no vote for $[d \succ e \succ \text{Others} \succ c]$ is eliminated by the lottery, then the winner must be d, because no matter what the manipulator votes for, in the resulting majority graph either $e \to c$ with weight 1 or $d \to c$ with weight 1, and in cases where there is an edge $c \to d$, its weight must be 1 (we recall that $e \to c$ will be fixed before considering $c \to d$, due to the tie-breaking mechanism). Therefore, the only cases where c wins is when a vote of $[d \succ e \succ \text{Others} \succ c]$ is eliminated in the first round.

If the UCM$_1$ instance has a solution, denoted by V, then we let the manipulator vote for $[c \succ e \succ d \succ V]$. This makes c win with non-zero probability $(n/(3n-1))$, which is strictly larger than 0 (the probability c wins when the manipulator votes for W).

On the other hand, suppose the manipulator can cast a vote V' to make c win with non-zero probability. We have already argued that in the cases where c wins, a vote for $[d \succ e \succ \text{Others} \succ c]$ must be eliminated in the first round. In such cases c is the winner under ranked pairs if and only if (1) both d and e are ranked below c in V', and (2) the vote obtained from V' by removing d and e is a solution to the UCM$_1$ instance.

Therefore, it is NP-hard to compute a fixed or improving manipulation for LetThenRankedPairs, even with a single manipulator. □

Finally, we prove that LotThenCopeland and LotThenMaximin are both intractable to manipulate.

THEOREM 13. *With unweighted votes and an unbounded number of candidates, for even a single manipulator, fixed and improving manipulation are* NP*-hard for LotThenCopeland and LotThenMaximin.*

Proof sketch: The proof is similar to the proof of Theorem 8. For both rules, we use a profile P illustrated in Table 2 to show the reduction. In this proof, w.l.o.g. $(t-q)$ is even.

#	Votes
for each $j \leq t$	$d \succ e \succ S_j \succ c \succ$ Others
$(t-q)/2$	$c \succ d \succ e \succ \mathcal{V}$ Rev($\mathcal{V}$) $\succ c \succ e \succ d$
for each $i \leq 3q$, $(t+2-q)/2 - \Delta(v_i)$	$d \succ v_i \succ c \succ e \succ$ Others Rev(Others) $\succ v_i \succ e \succ c \succ d$
for each $i \leq 3q$, $(t+2-q)/2 - \Delta(v_i)$	$d \succ c \succ e \succ v_i \succ$ Others Rev(Others) $\succ e \succ v_i \succ c \succ d$

Table 2: The profile *P* for fixed or improving manipulation.

Let $k = |P| + 1 - q$. For the fixed manipulation problem, we let $p = 0$. We note that the manipulator can make c win with positive probability only if she ranks c in the top, and the votes eliminated in P corresponds to an exact cover of $\mathcal{V}$. For the improving manipulation problem, we let $W = [d \succ e \succ \mathcal{V} \succ c]$. It follows that if the manipulator's vote is W, then c wins with 0 probability. Therefore, there is an improving manipulation if and only if the fixed manipulation problem (with $p = 0$) has a solution. □

6. SAMPLING THE RUNOFF VOTERS

So far we have not discussed in details how to select the runoff voters. Of course if we only need to select k voters uniformly at random, then we can perform a naïve k-round sampling: in each round, a voter is drawn uniformly at random from the remaining voters, and is then removed from the list. However, it is more difficult to generate k voters with some non-uniform distribution. For example, different voters in a profile may have different voting power [20], and we may therefore want to generate the voters in the runoff according to this voting power. More precisely, we want to compute a probability distribution over all sets of k voters, and each time we randomly draw a set (of k voters) according to this distribution to meet some constraints. Let $\mathcal{M}$ denote the set of all $n \times k$ 0-1 matrices, in each of which the sum of each row is no more than 1 and the sum of each column is exactly 1. That is, $\mathcal{M} = \{(a_{(i,j)}) : a_{(i,j)} \in \{0,1\}, \forall i \leq n, \sum_j a_{(i,j)} \leq 1$ and $\forall j \leq k, \sum_i a_{(i,j)} = 1\}$. Each matrix in $\mathcal{M}$ represents a set of k voters. Formally, we define the sampling problem as follows.

DEFINITION 5. *In the* LOTSAMPLING *problem , we are given a natural number n (the number of initial voters), a natural number k (the number of runoff voters), and a vector of positive rational numbers $(p_1, \ldots, p_n)$ such that for any $j \leq n$, $0 \leq p_j \leq 1$ and $\sum_{j \leq n} p_j = k$. We are asked to compute a sample of k voters such that, and for every $j \leq n$, the probability that vote j is chosen is p_j.*

Using algorithms in [7]. we have the following corollary.

COROLLARY 4. LOTSAMPLING *can be solved in polynomial time.*

7. CONCLUSIONS

Our main computational complexity results are summarized in Table 3. The prevalence of computational intractable results in this table suggests that lot-based voting is worth further attention. This simple non-deterministic tweak to voting rules appears to provide considerable (worst-case) resistance to manipulation. There are many directions for future work in addition to the questions already raised. For instance, we could consider the computational complexity of EVALUATION for other lot-based voting rules. In particular, we conjecture that EVALUATION for LotThenPlurality is NP-hard. We also intend to look at the cost of computing manipulations of

Rule X	LotThen$X$		
	EVALUATION	Fixed Manipulation	Improving Manipulation
Borda	NP-hard (Theorem 6)	NP-hard (Theorem 9)	NP-hard (Theorem 10)
STV	?	NP-hard (Theorem 11)	
Ranked pairs	NP-hard (Theorem 8)	NP-hard (Theorem 12)	
Copeland	NP-hard (Corollary 1)	NP-hard (Theorem 13)	
Maximin			

Table 3: Summary of our complexity results.

lot-based voting rules in practice [24, 25, 27]. We could also consider the control of lot-based voting by the chair. In addition to the usual forms of control like addition of candidates or of voters, we have another interesting type of control where the chair influences the outcome of the lottery. Such control is closely related to control by deletion of voters. Other types of control include the chair choosing the size of the lottery and the chair choosing the voting rule used in the runoff after the lottery. Another interesting direction would be to consider the computation of possible and necessary winners for lot-based voting rules [18, 26].

Acknowledgements

Toby Walsh is supported by the Australian Department of Broadband, Communications and the Digital Economy, the ARC, and the Asian Office of Aerospace Research and Development (AOARD-104123). Lirong Xia is supported by NSF under Grant #1136996 to the Computing Research Association for the CIFellows Project. We thank Ronald Rivest, anonymous WSCAI and AAMAS reviewers for very helpful suggestions and comments.

8. REFERENCES

[1] Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. In *Proc. STOC 2005*, pages 684–693, 2005.

[2] Noga Alon. Ranking tournaments. *SIAM Journal of Discrete Mathematics*, 20:137–142, 2006.

[3] John Bartholdi, III and James Orlin. Single transferable vote resists strategic voting. *Social Choice and Welfare*, 8(4):341–354, 1991.

[4] John Bartholdi, III, Craig Tovey, and Michael Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6:157–165, 1989.

[5] John Bartholdi, III, Craig Tovey, and Michael Trick. How hard is it to control an election? *Math. Comput. Modelling*, 16(8-9):27–40, 1992. Formal theories of politics, II.

[6] Garrett Birkhoff. Tres observaciones sobre el algebra lineal. *Univ. Nac. Tucumn Rev, Ser. A, no. 5*, pages 147–151, 1946.

[7] Min-Te Chao. A general-purpose unequal probability sampling plan. *Biometrika*, 69(3):653–656, 1982.

[8] Vincent Conitzer and Tuomas Sandholm. Universal voting protocol tweaks to make manipulation hard. In *Proc. 18th IJCAI*, pages 781–788, 2003.

[9] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proc. 7th ACM Conference on Electronic Commerce*, pages 82–90, 2006.

[10] Vincent Conitzer, Tuomas Sandholm, and Jérôme Lang. When are elections with few candidates hard to manipulate? *JACM*, 54(3):1–33, 2007.

[11] Oliver Dowlen. Sorting out sortition: A perspective on the random selection of political officers. *Political Studies*, 57:298–315, 2009.

[12] Edith Elkind and Helger Lipmaa. Hybrid voting protocols and hardness of manipulation. In *Proc. 16th ISAAC*, pages 206–215, 2005.

[13] Piotr Faliszewski, Edith Hemaspaandra, and Lane A. Hemaspaandra. Using complexity to protect elections. *Commun. ACM*, 53:74–82, 2010.

[14] Piotr Faliszewski, Edith Hemaspaandra, and Lane A. Hemaspaandra. Multimode control attacks on elections. *Journal of Artificial Intelligence Research*, 40:305–351, 2011.

[15] Piotr Faliszewski and Ariel D. Procaccia. AI's war on manipulation: Are we winning? *AI Magazine*, 31(4):53–64, 2010.

[16] Michael Garey and David Johnson. *Computers and Intractability*. W. H. Freeman and Company, 1979.

[17] Allan Gibbard. Manipulation of schemes that mix voting with chance. *Econometrica*, 45:665–681, 1977.

[18] Kathrin Konczak and Jérôme Lang. Voting procedures with incomplete preferences. In *Proc. IJCAI-2005 Workshop on Advances in Preference Handling*, 2005.

[19] Miranda Mowbray and Dieter Gollmann. Electing the Doge of Venice: Analysis of a 13th century protocol. In *Proc. IEEE Symposium on Computer Security Foundations*, pages 295–310, 2007.

[20] David M. Pennock and Lirong Xia. Voting power, hierarchical pivotal sets, and random dictatorships. In *Proc. IJCAI Workshop on Social Choice and Artificial Intelligence*, 2011.

[21] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proc. STOC 1978*, pages 216–226, 1978.

[22] Krzysztof Wojtas and Piotr Faliszewski. Possible winners in noisy elections. In *Proc. IJCAI Workshop on Social Choice and Artificial Intelligence*, 2011.

[23] Lirong Xia, Michael Zuckerman, Ariel D. Procaccia, Vincent Conitzer, and Jeffrey Rosenschein. Complexity of unweighted coalitional manipulation under some common voting rules. In *Proc. 21st IJCAI*, pages 348–353, 2009.

[24] Toby Walsh. Where Are the Really Hard Manipulation Problems? The Phase Transition in Manipulating the Veto Rule. In *Proc. 21st IJCAI*, pages 324–329, 2009.

[25] Toby Walsh. An Empirical Study of the Manipulability of Single Transferable Voting. In *Proc. 19th ECAI*, pages 257–262, 2010.

[26] Toby Walsh. Uncertainty in preference elicitation and aggregation. In *Proc. 22nd AAAI*, pages 3–8, 2007.

[27] Toby Walsh. Where Are the Hard Manipulation Problems? *Journal of AI Research*, 42: 1–39, 2011.

Convergence of Iterative Voting

Omer Lev
omerl@cs.huji.ac.il

Jeffrey S. Rosenschein
jeff@cs.huji.ac.il

School of Computer Science and Engineering
The Hebrew University of Jerusalem
Jerusalem 91904, Israel

ABSTRACT

In multiagent systems, social choice functions can help aggregate the distinct preferences that agents have over alternatives, enabling them to settle on a single choice. Despite the basic manipulability of all reasonable voting systems, it would still be desirable to find ways to reach a *stable* result, i.e., a situation where no agent would wish to change its vote. One possibility is an iterative process in which, after everyone initially votes, participants may change their votes, one voter at a time. This technique, explored in previous work, converges to a Nash equilibrium when Plurality voting is used, along with a tie-breaking rule that chooses a winner according to a linear order of preferences over candidates.

In this paper, we both consider limitations of the iterative voting method, as well as expanding upon it. We demonstrate the significance of tie-breaking rules, showing that when using a general tie-breaking rule, no scoring rule (nor Maximin) need iteratively converge. However, using a restricted tie-breaking rule (such as the linear order rule used in previous work) does not by itself *ensure* convergence. We demonstrate that many scoring rules (such as Borda) need not converge, regardless of the tie-breaking rule. On a more encouraging note, we prove that Iterative Veto does converge—but that voting rules "between" Plurality and Veto, k-approval rules, do not.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Economics, Theory

Keywords

Social choice theory, Iterative Voting, Nash Equilibrium

1. INTRODUCTION

When multiple agents have independent, perhaps differing, views over a set of alternatives, one way to decide upon an alternative is to use *social choice theory* (i.e., voting) to aggregate their preferences and arrive at a common decision. However, the possibility of strategic manipulation remains a potential pitfall; the well-known Gibbard-Satterthwaite result [7, 13] states that strategic voting is potentially beneficial in any reasonable non-dictatorial voting system. Hence, analyzing elections has been complicated by the possibility that voters would attempt manipulations, and/or speculate on the actions of other players so as to try and counter-manipulate. Further complicating analysis is that effective manipulation is strongly tied to the information each player has of the game and his knowledge of the truthful preferences of other players [15]; many papers dealing with manipulation assume that all players have complete information of the game.

One approach to understanding an election is to treat it as a process, and see if we can reach some point of equilibrium, where all players are satisfied with their votes/manipulations, no longer wishing to change them. The most obvious candidate for such a stable solution would be to find a *Nash equilibrium*. However, as there may be multiple Nash equilibria in a game, many of them trivial (e.g., when all voters vote for any specific candidate), this may seem like an overly-weak option to pursue. It is, in a sense, too broad a tool to use in analyzing an election.

In previous work, Meir et al. [10] suggested the framework of iterative voting: all participants vote, and then—knowing only the result—may change their votes, one at a time, though not in a predetermined order.[1] This iterative process stops when an equilibrium is reached, when no player wishes to change his vote. A similar process can be seen, "in action", online at various websites used to coordinate dates for an event, such as www.doodle.com; following an initial vote, every participant can change his vote. Obviously, as players change their choices one at a time, iterative voting rules are more naturally suited to a relatively small number of players, or an especially close election.

In their paper, Meir et al. [10] proved that using the simple Plurality voting rule, with a deterministic tie-breaking rule that uses a fixed linear order on candidates to break ties (and further assuming that all voters have equal weight), an iterative vote will converge to a Nash equilibrium when voters always give the best response possible to the current situation (in light of their preferences). They also showed that with weighted voters, or when using better-reply strategies (instead of best-replies), convergence is not guaranteed.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

[1]If they were allowed to vote simultaneously, it is easy to prove that the result may never converge to an equilibrium; and a predetermined order would just be a new voting rule.

The authors further explored nondeterministic tie-breaking rules, and showed that while they may not always converge, if the starting point is a truthful state and voters are unweighted, the game will converge.

In the current paper, we examine the robustness of this framework, as well as expanding it to encompass, beyond Plurality, an additional voting rule. We discover that when dealing with deterministic tie-breaking rules, the *type* of tie-breaking rule is crucial for a positive result: if we do not restrict the choice of tie-breaking rules, no scoring rule can guarantee convergence, and going beyond scoring rules, the Maximin voting rule is also not guaranteed to converge. Furthermore, regardless of the tie-breaking rule used, iterative voting cannot be generalized to all scoring rules, as the Borda voting rule is not guaranteed to converge under *any* tie-breaking rule. However, when using a linear-order tie-breaking rule, the iterative process with the Veto voting rule does converge when voters use the best-response strategy.[2] Examining if voting rules "between" Plurality and Veto (k-approval rules) are guaranteed to converge as well, we find that they are not.

1.1 Related Work

While we use the framework established by Meir et al. [10], the notions of an iterative approach to voting, as well as of seeking election equilibria, exist in previous research. An iterative process for reaching decisions was offered for agents in Ephrati and Rosenschein [5], but it uses a mechanism to transfer money-like value among agents, and hence is irrelevant to our voting procedures. Several researchers have considered reaching an equilibrium with an iterative (or dynamic) process, in particular when deciding on an allocation of public goods. A summary of much of that work can be found in Laffont [9], which details various approaches, including different equilibria choices (Nash, local dominant, local maximin) and methods. However, in order to reach an equilibrium, they limit the possible preference choices to single-peaked preferences. Another branch of research deals with a process of having a player propose a change in the current state, and hold a vote on its acceptance. Such a model was used by Shepsle [14], who chose to force an equilibrium by using a combination of preference limitation and organizational limitations. De Trenqualye [3] chose to achieve an equilibrium by using a specific voting rule and Euclidean preferences. More recently, Airiau and Endriss [1] examined—theoretically and experimentally—the possibility of an equilibrium in such games, using Plurality-type voting rules (the threshold can be different than 50% for a change to be accepted).

In searching for equilibria (albeit not iteratively), Feddersen et al. [6] chose (like Laffont) to limit preferences to single-peaked preferences. Others, like Hinich et al. [8], for example, chose to change the single-peak limitation to a specific probabilistic model of voters over a Euclidean space of candidates, while changing other parts of the model (such as allowing for abstentions). A somewhat different approach, taken by Messner and Polborn [11], analyzed equilibria by coalitional manipulation (hence, using a stronger equilibrium than Nash—a method also utilized by Dhillon and Lockwood [4]). However, one of the main limitations of many of the papers mentioned above is that they assume some knowledge of other players' preferences.

Attempting to investigate the role of knowing other players' knowledge, Chopra et al. [2] examined iterative voting with Plurality, and showed the effects of limiting a player's knowledge of the other players' preferences. Another interesting model, proposed in Myerson and Weber [12], found a Nash equilibrium for scoring rules, assuming that voters have some knowledge of which candidates have a better chance of winning (based, for example, on pre-election polls), but this does not mean that every election results in an equilibrium.

1.2 Overview of the Paper

In the following section, we give a brief overview of elections, and describe the model of iterative voting that we will be exploring throughout the paper. In Section 3 we show that the characteristics of the tie-breaking rule can affect convergence; we give examples showing that for every scoring rule, as well as for Maximin, there is a (non-linear-ordered) tie-breaking rule for which it will not always converge.

We show in Section 4 that the Borda voting rule is not guaranteed to converge, regardless of the tie-breaking rule used. After that, we limit ourselves to linear-ordered tie-breaking rules, and in Section 5 we show that using the Veto voting rule to create the Iterative Veto procedure results in a voting rule that always converges to a Nash equilibrium. However, we also show that generally, when using k-approval voting rules, they do not always converge, even when using linear-ordered tie-breaking rules. Finally, we discuss various open problems, and the issues that make them difficult (and interesting).

2. DEFINITIONS

2.1 Elections and Voting Systems

Before detailing our iterative game, we first define elections, and how winners are determined.

Definition 1. Let C be a group of m candidates, and let A be the group of all possible preference orders over C. Let V be a group of n voters, and every voter $v_i \in V$ has some element in A which is its true, "real" value (which we shall mark as a_i), and some element of A which it announces as its value, which we shall denote as $\tilde{a}_i$.

Note that our definition of a voter incorporates the possibility of its announcing a value different than its true value (strategic voting).

Definition 2. A voting rule is a function $f : A^n \to 2^C \setminus \emptyset$.

There are many known voting systems; one category among them is the family of scoring rules.

Definition 3. A scoring rule is a voting rule that uses a vector $(\alpha_1, \alpha_2, \ldots, \alpha_{m-1}, 0) \in \mathbb{N}^m$ such that $\alpha_i \geq \alpha_{i+1}$. Each voter gives α_1 points to its first choice, α_2 points to the second, and so on. The candidates with the highest scores are the winners.

[2]Iterative vetoing is used, in the real world, in various situations, such as elimination decisions in various "reality shows" (e.g., American Idol, America's Next Top Model, etc.). As they usually use a single judge's preferences to break ties, it is indeed linear-order tie-breaking.

There are several well-known scoring rules.

- **Plurality**: The scoring vector is $(1, 0, 0, \ldots, 0)$—a point is only given to the most preferred candidate.
- **Veto**: The scoring vector is $(1, 1, 1, \ldots, 1, 0)$—a point is given to everyone except the least-preferred candidate.
- **Borda**: The scoring vector is $(m-1, m-2, \ldots, 2, 1, 0)$—a candidate receives points according to its preference rank.
- **k-approval (or k-veto)**: The scoring vector is $(1, 1, \ldots, 1, 0, 0, \ldots, 0)$—a point is given to the most preferred k candidates (or points are given to all except the least-preferred k candidates).

There are also voting systems that are not scoring rules, such as Maximin.

Definition 4. The Maximin voting rule defines for every two candidates $x, y \in V$ a score $N(x, y)$ which is the number of voters who preferred x over y. Each candidate then receives the score $S_x = \min_{y \in V \setminus x} N(x, y)$. The winners are the candidates with the maximal score.

Our definition of voting rules allows for multiple winners. However, in many cases what is desired is a single winner; in these cases, a tie-breaking rule is required.

Definition 5. A tie-breaking rule is a function $t : 2^C \to C$ that, given a set of elements in C, chooses one of them as a (unique) winner.

There can be many types of tie-breaking rules, such as random or deterministic, lexical or arbitrary. One family of tie-breaking rules that will be of interest to us is the family of linear-ordered tie-breaking rules.

Definition 6. Linear-ordered tie-breaking rules are tie-breaking rules that decide upon a winner based on some preference order over C (an element of A). Practically, this means that if $a, b \in D \subseteq C$ and $t(D) = a$, then if $a, b \in D' \subseteq C$, then $t(D') \neq b$.

While this paper does not deal with weighted games, expanding the above definitions to games that allow weighted voters is straightforward: a voter v_i with weight w_i is considered as if it were w_i different voters with the same preferences and strategy.

2.2 The Iterative Game

The following definitions and explanations follow the framework established by [10]. We do not assume that every voter knows the preferences of the others; on the contrary, we assume that each player only knows the current results (and scores) of the game, and is not aware of other voters' preferences. Hence, voters are myopic; they only think of changing their vote so as to improve the current situation, as they do not take into account future steps by other players (we also assume they are not trying to learn what their rivals' preferences are, based on their strategies).

Formally, we are viewing the election as a game, in which each player has an internal preference (a_i), and a strategy $(\tilde{a}_i)$. The outcome of the game is $t(f(\tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_n))$. We wish to find a Nash equilibrium, in which no player wishes to change his strategy, i.e., a situation in which, for any voter $v_i \in V$ and any $a_i' \in A$:

$$t(f(\tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_i, \ldots, \tilde{a}_n)) \succeq_{a_i} t(f(\tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_i', \ldots, \tilde{a}_n))$$

However, we do not just want to prove that such an equilibrium exists; rather, we wish to show a process that makes that equilibrium reachable from the original starting point (which in many cases might be, due to lack of prior information, truthful on the part of the voting agents—though this is not a necessary requirement for our proofs).

Definition 7. An iterative election game G is made up of an initial election (which we shall mark as G_0), followed by further elections $(G_1, G_2, \ldots)$, with the difference between election G_i and G_{i+1} being that one voter changed his declared preference. A game is stable if there is an n such that for all $i > n$, $G_i = G_n$.

G_0 may be a truthful state (i.e., voters vote according to their real preferences), but it is not necessarily so.

Obviously, since at every step some voter may change something about his reported preferences, no election need be stable. However, analysis becomes more interesting once we limit the voter's possible changes, requiring individual rationality. In that case, a valid step is one in which the winner of the election changes (due to the myopic steps of the players, a move that does not change the winner is pointless), and one which changes the winner to one that the voter who changed his strategy finds more preferred than the previous winner (according to the voter's internal, real preferences). Formally, a step from G_i to G_{i+1} is one in which all voters played in G_i according to the strategies $(\tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_n) \in A^n$, and for some player $v_j \in V$, there is a strategy $\tilde{a}_j' \in A$ such that:

$$t(f(\tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_j', \ldots, \tilde{a}_n)) \succ_{a_j} t(f(\tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_j, \ldots, \tilde{a}_n))$$

Under the assumption of individual rationality, a stable game is one that has reached a Nash equilibrium.

Furthermore, we define a specific type of step for our analysis, following [10].

Definition 8. A *best response* step is one in which the voter changing his strategy cannot cause a more preferred candidate to win using a different strategy.

In specific voting rules below, we shall further refine what a best response move means in various circumstances.

3. USING ARBITRARY TIE-BREAKING RULES

3.1 Scoring Rules

THEOREM 1. *An iterative scoring rule game with a deterministic tie-breaking rule, even for unweighted voters that use best-response moves and start from a truthful state, does not always converge.*

PROOF. Our examples will be somewhat complex, as we deal with a large family of voting rules. In some cases using a best response is obvious, as there is only one choice that results in making a specific candidate the winner. In other cases, there may be multiple options to reach the same winner. In the examples below, we use a "natural" definition, in which players who are taking off points from the current winner will a) give him 0 points, b) award the new winner maximal points, and c) if all things are equal, will prefer to be as close as possible to their truthful preferences. However, even if one does not use such a definition, cycles are still created—just longer cycles, as they may go through several more steps than detailed here.

First, we shall deal with the case where there are at least three candidates that do not receive maximal scores (i.e., $\alpha_{m-2}, \alpha_{m-1}, \alpha_m < \alpha_1$). We have at least four candidates, a, b, c and d. Our tie-breaking rule is that when c is tied with others, except b, c wins. When b is tied with others, except d, b wins. When d is tied with others, except a and c, d wins. We write $a \succ b \succ c$ to express that a is a voter's favorite candidate, c is his least preferred candidate, and b is ranked in between. We have two voters:

Voter 1: $a \succ b \succ c \succ d$
Voter 2: $c \succ d \succ b \succ a$

We can add several dummy candidates so the score given by voter 1 to b is less than is given to a, and the score voter 2 gives to d is less than given to c (and dummy voters, making these dummy candidates irrelevant as winner possibilities). The winner in this state is c (either he is the sole winner, or through a tie with a). The only option for improving the result for voter 1 is to make b victorious, changing his preference to $b \succ a \succ d \succ c$. Voter 2 can improve the result by changing his preference to $d \succ c \succ a \succ b$, making d the winner (possibly through winning the tie between b and d). The option available to voter 1 is to return to his original preference order, making a, his favorite, the winner. However, now voter 2 will return to his original preference as well, as it ensures the victory of c, his most preferred candidate.

If there are only two candidates that receive less than the maximal score, then we use a different game, one with six candidates. Our tie-breaking rule makes b win when a, b, c, d are tied; a wins when a, c, d are tied; c wins when a and c are tied; and c wins when a, c, d, e are tied (other ties that include d make him the winner; if they do not include d, but do include e or f, then e/f is the winner, with f triumphing over e). Let us look at two voters:

Voter 1: $a \succ b \succ c \succ d \succ e \succ f$
Voter 2: $b \succ c \succ a \succ d \succ e \succ f$

The winner here is candidate b (since a, b, c, d are tied). However, when voter 1 changes his stated preference to $a \succ c \succ d \succ e \succ f \succ b$, then a, his favorite, becomes the winner (since a, c, d are tied). Voter 2 can only improve this situation by changing his stated preference to $a \succ c \succ d \succ e \succ b \succ f$, making c victorious. Voter 1 can now improve his situation by returning to his original preference, making a the winner. In this case, voter 2 will gladly return as well to his original preference, as that will make his favorite candidate, b, win.

If there is only one candidate that receives the less-than-maximal score, this is the Veto voting rule, for which there is a similar, but simpler, example. We shall use two voters, and we can describe the voting rule and tie-breaking rule fully with a table, marking the victor according to whom the voters chose to veto.

	a	**b**	**c**	**d**	**e**
a	b	c	d	e	d
b	c	d	a	a	a
c	d	a	b	a	a
d	e	a	a	a	a
e	d	a	a	a	a

In our case, the voters' real preferences are:

Voter 1: $c \succ b \succ d \succ e \succ a$
Voter 2: $b \succ d \succ c \succ e \succ a$

The truthful starting point would result in b being the winner. As voter 1 would rather that c win, he will move to veto b. Following that, voter 2 would move to veto b as well, as that would result in d winning. Voter 1, who would rather that c win, will return to vetoing a, and as voter 2 would rather that b be victorious, would return to vetoing a as well, returning to our original starting point. □

3.2 Maximin

Theorem 2. *An iterative Maximin game with deterministic tie-breaking, even for unweighted voters that use best-response moves and start from a truthful state, does not always converge.*

Proof. We shall again use two voters and four candidates. The voters' preferences are:

Voter 1: $c \succ d \succ b \succ a$
Voter 2: $b \succ d \succ c \succ a$

We define the tie-breaking rule as follows: $b = c = d \Rightarrow b$; $c = b \Rightarrow c$; $a = b = c \Rightarrow b$; $a = b = c = d \Rightarrow c$; $c = d \Rightarrow c$; $b = d \Rightarrow b$. All the rest include a, and result in a being the winner.

Beginning in a truthful state, the scores of c, b and d are tied at the top, hence b is the winner. Voter 1 has no better manipulation than one that makes c victorious, and changes his preference to $c \succ b \succ d \succ a$, which evens the score of b and c, and c is the winner. Voter 2 now seeks to make b the winner, and succeeds by announcing his preferences to be $a \succ b \succ d \succ c$ (which ties, with the top score, a, b, and c). Voter 1, by returning to his original preference list, makes the score of a, b, c and d equal, resulting in c being the winner, and Voter 2 can retaliate by returning to his original preference list as well, under which b, his favorite, was victorious. □

4. USING BORDA

Despite the significance of tie-breaking rules, there are voting rules that will not converge, regardless of the tie-breaking rule used.

Theorem 3. *An iterative Borda game with every type of tie-breaking rule, even for unweighted voters that use best-response moves and start from a truthful state, does not always converge.*

Proof. The example here is the same as the first example used in the proof of Theorem 1. However, the analysis in the Borda rule is much simpler, as in this case ties never occur, and hence, there is no need to rely on tie-breaking rules to achieve the necessary result: at every stage the winner will

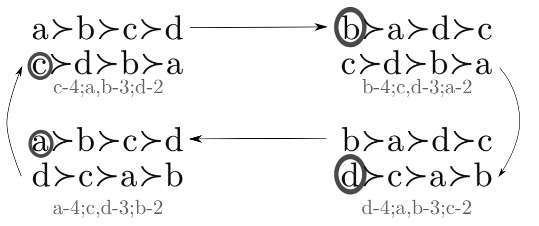


Figure 1: The cycle of Borda non-convergence (top left is the truthful state)

have 4 points, while the other candidates will have 3 points or fewer (see Figure 1).

□

Notice that this proof stands for all scoring rules for which $m \geq 4$ and for which $\alpha_1 > \alpha_2$ and $\alpha_3 > 0$, and Borda is just one example of such a scoring rule.

5. USING VETO AND K-APPROVAL

If we confine our work to tie-breaking rules that enforce a linear order on candidates, most of our counter-examples no longer work, and convergence becomes a possibility.

5.1 Veto

Definition 9. A *best response* in the case of the Veto voting rule implies that the current (undesired) winner is vetoed.

THEOREM 4. *An iterative Veto game with deterministic linear-order tie-breaking and unweighted voters which use a best-response strategy, converges even when not starting from a truthful state.*

PROOF. Suppose there is an iterative election game G that includes a cycle. We shall confine our game to the cycle only, mark an arbitrary state in the cycle as G_0, and enumerate the rest of the cycle accordingly. Note that G_0 is not necessarily the opening state of the original game.

Definition 10. $score_i(x)$ is defined as the score of candidate x in game state G_i. $max(G_i)$ is defined as the score of the winning candidate in G_i.

LEMMA 5. *If there is a cycle, then for $j < i$, $max(G_i) \leq max(G_j) + 1$, and if $max(G_i) = max(G_j) + 1$, there is only one candidate with that score.*

PROOF. By induction: for $n = 0$ the lemma is true by definition. Assuming it is true for $n' < n$, we shall prove it for n. If for some j, $max(G_{n-1}) = max(G_j) + 1$, then it is a single candidate, and therefore, the next stage will make that candidate's score go down to $max(G_j)$, and add a point to another candidate. As that candidate's score was less than $max(G_{n-1})$, its new score will be, at most, $max(G_{n-1})$, and if it is exactly $max(G_{n-1})$, it is a single candidate. If $max(G_n) < max(G_{n-1})$, then due to the induction assumption, $max(G_n) < max(G_i) + 1$ for $i < n$ (there cannot be equality, for that means $max(G_{n-1}) > max(G_i) + 1$).

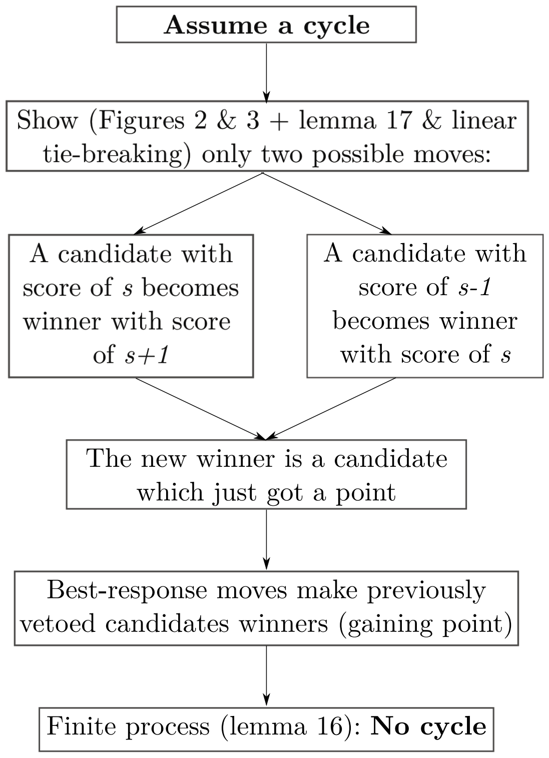

Assume a cycle
Show (Figures 2 & 3 + lemma 17 & linear tie-breaking) only two possible moves:
A candidate with score of s becomes winner with score of s+1
A candidate with score of s-1 becomes winner with score of s
The new winner is a candidate which just got a point
Best-response moves make previously vetoed candidates winners (gaining point)
Finite process (lemma 16): No cycle

Figure 2: General overview of Veto proof

If, for every $i < n$, $max(G_{n-1}) \leq max(G_i)$, then if the candidate that gets an additional point at stage n has a lower score than $max(G_{n-1})$, this means $max(G_n) \leq max(G_{n-1})$, and the induction requirements still stand. If it has the score $max(G_{n-1})$, it becomes the single candidate with a score of $max(G_{n-1}) + 1$, and $max(G_n) \leq max(G_i) + 1$ for $i < n$. □

Notice that since we can choose G_0 arbitrarily from the cycle, due to the last lemma, $max(G_0) + 1 \geq max(G_i) \geq max(G_0) - 1$, otherwise, there will be no possibility for the cycle to return to its starting point.

LEMMA 6. *There can be at most $n \cdot (m - 2)$ consecutive steps in which the voter changed his veto from candidate a to candidate b, and candidate a became the winner.*

PROOF. Every time a voter changes his veto, he indicates that he prefers the current vetoed candidate to the current winner; that is, the winner is someone he likes less and less as the game progresses. Since there are n voters and, at most, $m-1$ candidates that are worse than the current one, and as the voter will not vote for the very worst candidate, there are $n \cdot (m - 2)$ steps. □

We shall deal, first of all, with the easiest case, solved by the lemma above, when there is always only one candidate with the winning score (the tie-breaking rule is never used). In this case, at every step, the old winner loses a point, and the new winner gets a point. This is the case dealt with in Lemma 6, and as the number of steps is limited, there can be no cycle.

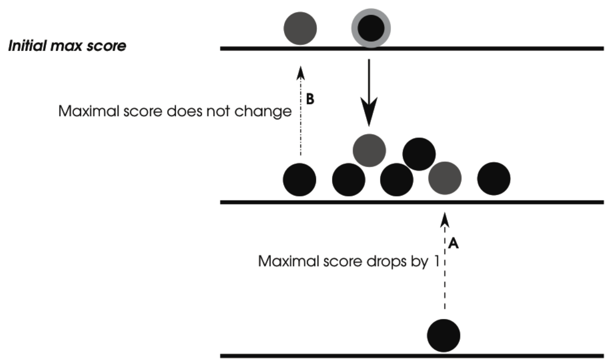


Figure 3: When only one candidate has maximal score

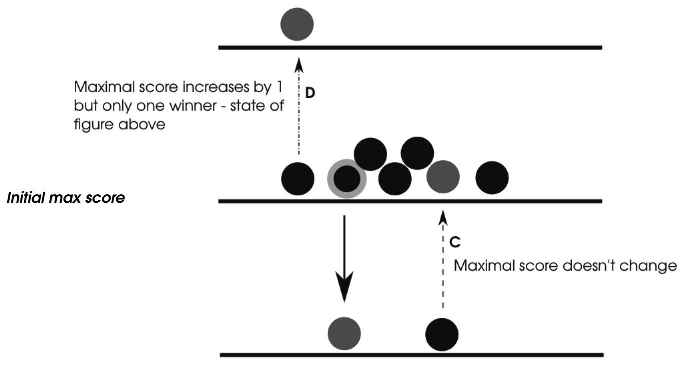

Maximal score increases by 1 but only one winner - state of figure above
D
Initial max score
C
Maximal score doesn't change

Figure 4: When multiple candidates have the maximal score

Diagrams showing why there is a limit on the increase and decrease of maximal score. When there is a state with only one candidate with maximal score, the maximal score will either remain the same with a single winner (move type A) or decrease (move type B). If it is a state where there are several candidates with the maximal score, the maximal score will either (move type D) increase while creating a situation with a single winner (which cannot increase) or the maximal score will remain the same (move type C). This also illustrates why the score cannot go down much if there is a cycle—it can only increase by one in the whole cycle; at no point can we reach a maximal score 2 points higher than another.

Having dealt with that case, let us take a closer look at G_0, which we can define as one of the states in which there is more than one candidate with a maximal score. Note that there must be more than one of these states, since if there were a single winner in G_i and more than that in G_{i+1}, a candidate received a point and did not become a unique winner, i.e., his score in G_i was, at most, $max(G_i)-2$. Since this is a cycle, there must be a step in which he returns to that score (if it is G_{i+1}, then for a cycle to happen, the same candidate will need to rise again so the voter that increased his score in G_i will veto him again).

LEMMA 7. *For every state G_i in which there is more than one candidate scoring $max(G_i)$, $max(G_i) = max(G_0)$, $|\{x|score_i(x) = max(G_i)\}| = |\{x|score_0(x) = max(G_0)\}|$ and $|\{x|score_i(x) = max(G_i)-1\}| = |\{x|score_0(x) = max(G_0)-1\}|$. This means there is always the same number of candidates with the maximal score, and with maximal score -1. Furthermore, these are always the same candidates, switching between the two scores: $\{x|score_i(x) \in \{max(G_i), max(G_i)-1\}\} = \{x|score_0(x) \in \{max(G_0), max(G_0)-1\}\}$.*

PROOF. According to Lemma 5, if $max(G_i) = max(G_0)+1$, there is only one candidate with the winning score, and we are not dealing with such a state. Suppose $max(G_i) = max(G_0)-1$; according to the same lemma, this means there is only one candidate with the winning score in G_0, which we defined as a state having at least two.

At any step in the game, one player loses a point and another gets it. Hence, if the number of those with the maximal score and maximal -1 score is not the same as in G_0, some candidate lost (or gained) a point, which has a score lower than maximal -1. However, as the maximal score will never be $max(G_0) - 1$ (otherwise, according to Lemma 5, there would only be one candidate with winning score in G_0), there is no way in the cycle for the candidate to be vetoed when it has a score of $max(G_0) - 1$, and get a lower score. As no candidate that has a score of $max(G_0)$ or $max(G_0)-1$ can get a smaller score, the group of candidates with these scores stays fixed throughout the cycle. □

Definition 11. Let B be the group of candidates who changed places in states in which $max(G_i) = max(G_0)$: $\{x|\exists i$ such that $score_i(x) = max(G_0)$ and $\exists j$ such that $score_j(x) = max(G_0) - 1\}$.

Let $z \in B$ be the lowest ranked candidate, according to the linear tie-breaking rule, in B. However, as at some state i it has a score of $max(G_0)$, and at state j has a score of $max(G_0) - 1$, there is a state i' in which it is vetoed and its score drops. At state i' it is the winner, meaning that all $b \in B \setminus z$ have a score of $max(G_0) - 1$. This further means that there is always only one element of B with the score of $max(G_0)$ (since the number of candidates with that score is constant, and candidates not in B never have a score of $max(G_0) - 1$), and that candidate is always victorious over any other candidates with that score (since it is a part of B, it needs to be vetoed).

Thus a step during the game will either give a candidate with a score of $max(G_0)$ a point that will make him win by giving him a score of $max(G_0) + 1$, or give a point to a candidate with the score of $max(G_0) - 1$ (and veto the single candidate with a score of $max(G_0) + 1$, or, if such does not exist, a candidate in B with a score of $max(G_0)$), making him the winner. Hence, the voted-for candidates always become the winners, and according to Lemma 6, this is a finite process.

5.2 k-Approval, $k \geq 3$

For the k-approval voting rule, for $k \geq 3$ we prove, as for Borda, a stronger claim than for general scoring rules—we prove that even when using linear-ordered tie-breaking rules, k-approval is not guaranteed to converge.

Theorem 8. *An iterative k-approval or k-veto game, when $k \geq 3$, with linear-ordered tie-breaking rule, even for unweighted voters that use best-response moves and start from a truthful state, does not always converge.*

Proof. We shall provide a proof for 3-approval—it can be expanded for any larger k by adding additional dummy variables. Our tie-breaking rule is linear, with the preference: $b \succ c \succ a \succ d \succ e \succ f$. Let us assume the existence of 50 voters whose preferences are $a \succ b \succ c \succ d \succ e \succ f$. 50 others prefer $a \succ b \succ d \succ c \succ e \succ f$; 50 others prefer $b \succ c \succ d \succ a \succ e \succ f$, and 50 others $a \succ c \succ d \succ b \succ e \succ f$. So a, b, c and d have the same number of points, which is maximal—150, and e and f have 0 points. Another voter votes for $a \succ d \succ e \succ b \succ c \succ f$, and the 3 voters we will deal with have the following preferences:

Voter 1: $b \succ a \succ e \succ f \succ c \succ d$
Voter 2: $c \succ b \succ e \succ f \succ d \succ a$
Voter 3: $d \succ c \succ a \succ e \succ f \succ b$

Following all voters, a is the winner with 153 points. b, c and d have 152 points each, e has 3 points, and f has no points. Voter 1 realizes that he can make his favorite, b, win, by changing his vote to $b \succ e \succ f \succ c \succ d \succ a$ (a, b, c and d are now all tied with 152 points). Voter 2 now sees he can make his favorite, c, win, by changing his vote to $c \succ e \succ f \succ d \succ a \succ b$ (a, c and d are tied with 152 points, b has 151 points). At this point, voter 3 realizes he too can make his favorite the victor, by changing his vote to $d \succ e \succ f \succ b \succ c \succ a$ (so d has 152 points, a, b and c have 151 points). Now, voter 1 understands that returning to his original vote would make a the winner, which he prefers over d (now a and d are tied with 152 points). Following that, voter 2 sees that he can make b victorious by returning to his previous preference (since a, b and d will be tied with 152 points). Returning to our starting position, voter 3 sees that returning to his original vote would make a the winner, which is preferable, for him, over b. $\square$

6. DISCUSSION AND FUTURE WORK

An iterative voting process has a certain natural attractiveness, allowing voters to modify their stated preferences, in light of what they see about the results of an election. Assuming that all voters are equivalently entitled to make modifications, it seems an appealing way to acknowledge the strategic nature of voters, allowing them to change their votes to get results they prefer. If the process converges, we reach some stable expression of the aggregated group preference—but the process may not converge.

We began by shedding some light on the limitations of this mechanism, and on some of the elements that enable it to converge, under specific circumstances. We showed that the makeup of the tie-breaking rule is critical for iterative voting to become a useful, converging mechanism. This is due to the basic construction of iterative voting; if ties never occur, the analysis is straightforward, either towards guaranteed convergence or not (Iterative Borda does not always converge; Iterative Plurality and Veto always will).

As ties are a significant element of what complicates the iterative convergence problem, the specific mechanism used to resolve them is part of what guarantees convergence (or lack of it): Some tie-breaking rules in certain circumstances ensure convergence; others do not. There is still much to clarify regarding this interaction between tie-breaking rules and equilibria. We have yet to establish the necessary requirements on tie-breaking rules that ensure convergence even when dealing with Iterative Plurality, let alone with other voting rules. We conjecture that requirements may be different for weighted and unweighted voting games.

However, even when eliminating considerations of tie-breaking rules, and even when we limit ourselves to scoring rules, we see that some voting rules—in fact, most of them—will never give us guaranteed convergence (such as Borda, and similar scoring rules with more than three different values). Furthermore, even if we allow ourselves to use only 1s and 0s in our scoring rules, we reach the surprising conclusion that other than the edges of this space (i.e., preference vectors where all but one element is 0, or all but one element is 1), almost no other part of this space can guarantee convergence.

This points to the basic difficulty of the iterative process—in many types of voting rules, a voter's change of stated preference may have unintended side-effects, so when a player wishes to make a certain candidate victorious, he may be inadvertently setting the stage for another candidate to become a viable contender, to be made the winner by another player. Contrast that with Iterative Plurality or Veto, in which only one candidate benefits (and as a corollary, only one candidate is damaged). However, this is still highly dependent on tie-breaking rules, and hence there might be some tie-breaking criteria that would enable these voting rules to converge as well.

The problem of unexpected candidates becoming viable is further exacerbated with the Maximin voting rule. In that case we encounter a problem of actually defining a "best response"—no longer is there a straightforward definition of a single candidate gaining or losing points. Rather, many candidates may be affected, but which ones will be affected and made viable in the long run cannot be computed or analyzed in a simple or predictable way. While we have begun to analyze this particular non-scoring-rule voting mechanism, the iterative process for these types of voting rules remains mostly unexplored.

The non-convergence of many voting rules may suggest that it would be useful to consider different solution strategies, instead of best-response. While we know [10] that using a better-response strategy does not help, as it will not guarantee convergence even for Iterative Plurality, other solution strategies may enable convergence for a wider range of voting rules. However, we have yet to find a satisfactory voting strategy, which is both natural and ensures convergence.

The new voting rule we explored, **Iterative Veto**, despite having some superficial resemblance to Iterative Plurality, does not have the "self-reinforcing" dynamic that Iterative Plurality has, in which once a candidate has become nonviable he will never return to relevance. In Iterative Veto, candidates' scores can increase very little from the initial stage, and when their score decreases, it may increase the number of viable candidates, making the process more turbulent then its Plurality equivalent. By making the ultimate winner potentially a candidate which was not viable at the outset, Iterative Veto enables us to reach Nash equilibria that were impossible using Iterative Plurality.

On a final note, we have not dealt with computational complexity issues here, as they were not relevant in the scenarios we considered. However, when expanding the analysis to other voting rules, such issues may arise. For example, [15] showed that finding a manipulation, even for a single manipulator in an unweighted game, is NP-complete for "ranked-pair" games such as STV and second-order Copeland. Therefore, each voter may struggle to find the step to improve his situation (and of course, struggle to find his best response).

7. ACKNOWLEDGMENTS

This work was partially supported by Israel Ministry of Science and Technology grant #3-6797, and the Google Inter-University Center for Electronic Markets and Auctions.

8. REFERENCES

[1] S. Airiau and U. Endriss. Iterated majority voting. In F. Rossi and A. Tsoukias, editors, *Algorithmic Decision Theory*, volume 5783 of *Lecture Notes in Computer Science*, pages 38–49. Springer Berlin / Heidelberg, 2009.

[2] S. Chopra, E. Pacuit, and R. Parikh. Knowledge-theoretic properties of strategic voting. In J. J. Alferes and J. Leite, editors, *Logics in Artificial Intelligence*, volume 3229 of *Lecture Notes in Computer Science*, pages 18–30. Springer Berlin / Heidelberg, 2004.

[3] P. de Trenqualye. An extension of Bowen's dynamic voting rule to many dimensions. *Social Choice and Welfare*, 15(1):141–159, 1998.

[4] A. Dhillon and B. Lockwood. When are plurality rule voting games dominance-solvable? *Games and Economic Behavior*, 46(1):55–75, January 2004.

[5] E. Ephrati and J. S. Rosenschein. Deriving consensus in multiagent systems. *Artificial Intelligence*, 87(1-2):21–74, November 1996.

[6] T. J. Feddersen, I. Sened, and S. G. Wright. Rational voting and candidate entry under plurality rule. *American Journal of Political Science*, 34(4):1005–1016, November 1990.

[7] A. Gibbard. Manipulation of voting schemes. *Econometrica*, 41(4):587–602, July 1973.

[8] M. J. Hinich, J. O. Ledyard, and P. C. Ordeshook. Nonvoting and the existence of equilibrium under majority rule. *Journal of Economic Theory*, 4(2):144–153, April 1972.

[9] J.-J. Laffont. Incentives and the allocation of public goods. In A. J. Auerbach and M. Feldstein, editors, *Handbook of Public Economics*, volume 2 of *Handbook of Public Economics*, chapter 10, pages 537–569. Elsevier, April 1987.

[10] R. Meir, M. Polukarov, J. S. Rosenschein, and N. R. Jennings. Convergence to equilibria of plurality voting. In *The Twenty-Fourth National Conference on Artificial Intelligence*, pages 823–828, Atlanta, Georgia, July 2010.

[11] M. Messner and M. K. Polborn. Strong and coalition-proof political equilibria under plurality and runoff rule. *International Journal of Game Theory*, 35(2):287–314, 2007.

[12] R. B. Myerson and R. J. Weber. A theory of voting equilibria. *The American Political Science Review*, 87(1):102–114, March 1993.

[13] M. A. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10(2):187–217, April 1975.

[14] K. A. Shepsle. Institutional arrangements and equilibrium in multidimensional voting models. *American Journal of Political Science*, 23(1):27–59, February 1979.

[15] L. Xia, M. Zuckerman, A. D. Procaccia, V. Conitzer, and J. S. Rosenschein. Complexity of unweighted coalitional manipulation under some common voting rules. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, IJCAI '09, pages 348–353, Pasadena, California, July 2009.

Optimal Manipulation of Voting Rules

Svetlana Obraztsova
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
SVET0001@ntu.edu.sg

Edith Elkind
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
eelkind@ntu.edu.sg

ABSTRACT

Complexity of voting manipulation is a prominent research topic in computational social choice. In this paper, we study the complexity of *optimal* manipulation, i.e., finding a manipulative vote that achieves the manipulator's goal yet deviates as little as possible from her true ranking. We study this problem for three natural notions of closeness, namely, swap distance, footrule distance, and maximum displacement distance, and a variety of voting rules, such as scoring rules, Bucklin, Copeland, and Maximin. For all three distances, we obtain poly-time algorithms for all scoring rules and Bucklin and hardness results for Copeland and Maximin.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Algorithms, Theory

Keywords

voting, manipulation, swap distance, footrule distance

1. INTRODUCTION

Mechanisms for aggregating the preferences of heterogeneous agents play an important role in the design of multi-agent systems [9]. Such mechanisms are typically implemented by *voting rules*, i.e., mappings that, given the rankings of the available alternatives by all agents, output an alternative that best reflects the collective opinion. There are many different voting rules that are used for group decision making; see, e.g., [3] for an overview.

A weakness shared by all reasonable voting rules is their susceptibility to *manipulation*: for any voting rule over a set of alternatives C, $|C| \geq 3$, that is not a dictatorship, there are voting situations where some voter would be better off if, instead of submitting her true ranking of the alternatives, she submitted a vote that did not quite match her true preferences. This was observed by Gibbard [11] and, independently, by Satterthwaite [18] more than 30 years ago, and a lot of research effort since then has been spent on identifying voting rules that are at least somewhat resistant to manipulation.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

In their pioneering paper [2], Bartholdi, Tovey and Trick proposed to use computational complexity as a roadblock in the way of manipulative behavior: they observed that, in practice, the manipulator needs an efficient method to find a successful manipulative vote, and a voting rule that does not admit such a method may be viewed as being relatively less vulnerable to manipulation. However, most classic voting rules, with the notable exception of STV, turn out to be susceptible to manipulation in this sense [2, 1].

In this paper, we study a refinement of the question asked by Bartholdi, Tovey and Trick. We observe that, while the manipulator is willing to lie about her preferences, she may nevertheless prefer to submit a vote that deviates as little as possible from her true ranking. Indeed, if voting is public (or if there is a risk of information leakage), and a voter's preferences are at least somewhat known to her friends and colleagues, she may be worried that voting non-truthfully can harm her reputation—yet hope that she will not be caught if her vote is sufficiently similar to her true ranking. Alternatively, a voter who is uncomfortable about manipulating an election for ethical reasons may find a lie more palatable if it does not require her to re-order more than a few candidates. Finally, a manipulator may want to express support for candidates she truly likes, even if these candidates have no chances of winning; while she may lie about her ranking, she would prefer to submit a vote where her most preferred candidates are ranked close to the top.

These scenarios suggest the following research question: does a voting rule admit an efficient algorithm for finding a manipulative vote that achieves the manipulator's goals, yet deviates from her true ranking as little as possible? To make this question precise, we need to decide how to measure the discrepancy between the manipulator's true preferences and her actual vote. Mathematically speaking, votes are permutations of the candidate set, and there are several *distances* on permutations that one can use. In our work, we consider what is arguably the two most prominent distances on votes, namely, the *swap distance* [12] (also known as bubble-sort distance, Kendall distance, etc.) and the *footrule distance* [20] (also known as the Spearman distance), as well as a natural variation of the footrule distance, which we call the *maximum displacement* distance.

In more detail, the swap distance counts the number of candidate pairs that are ranked differently in two preference orderings. Thus, when the manipulator chooses her vote based on the swap distance, she is trying to minimize the number of swaps needed to transform her true ranking into the manipulative vote. We remark that for swap distance, our problem can be viewed as a special case of the swap bribery problem [8]; however, our question is not addressed by existing complexity results for swap bribery [8, 7, 6, 19] (see

Section 7 for a discussion). The footrule distance and the maximum displacement distance are based on computing, for each candidate, the absolute difference between his positions in the two votes; the footrule distance then computes the sum of these quantities, over all candidates, while the maximum displacement distance returns the largest of them. We believe that each of these distances captures a reasonable approach to defining what it means for two votes to be close to each other; therefore, we are interested in analyzing the complexity of our manipulation problem for all of them.

We study our problem for several classic voting rules, namely, Bucklin, Copeland, Maximin, as well as all scoring rules. For all these rules, the algorithm of Bartholdi et al. [2] finds a successful manipulation if it exists. However, this algorithm does not necessarily produce a vote that is optimal with respect to any of our distance measures: in particular, it always ranks the manipulator's target candidate first, even if this is not necessary to achieve the manipulator's goal. Thus, we need to devise new algorithms—or prove that finding an optimal manipulation is computationally hard.

For all three distances, we obtain the same classification of these rules with respect to the complexity of finding an optimal manipulation: our problem is easy for Bucklin and all polynomial-time computable families of scoring rules (see Section 2 for definitions), but hard for Copeland and Maximin. For swap distance and footrule distance, we strengthen these hardness results to show that our problem is, in fact, hard to approximate up to a factor of $\Omega(\log m)$, where m is the number of candidates.

Our results provide a fairly complete picture of the complexity of finding an optimal manipulative vote for the three distances and four types of voting rules that we consider. Interestingly, they indicate that scoring rules (and the Bucklin rule, which is closely related to a subfamily of scoring rules known as k-approval) are fundamentally easier to manipulate than Copeland and Maximin; we remark that this observation is also suggested by the recent work of Obraztsova et al. [16, 15] on the complexity of manipulation under randomized tie-breaking. Thus, we believe that, besides being interesting for its own sake, our work contributes to the broad agenda of understanding the intrinsic complexity—and, therefore, practical applicability—of various voting rules.

2. PRELIMINARIES

An *election* is given by a set of candidates $C = \{c_1, \dots, c_m\}$ and a vector $\mathcal{R} = (R_1, \dots, R_n)$, where each R_i, $i = 1, \dots, n$, is a linear order over C; R_i is called the *preference order* (or, *vote*) of voter i. For readability, we will sometimes write $\succ_i$ in place of R_i. If $a \succ_i b$ for some $a, b \in C$, we say that voter i *prefers* a to b. We denote by $r(c_j, R_i)$ the *rank* of candidate c_j in the preference order R_i: $r(c_j, R_i) = |\{c \in C \mid c \succ_i c_j\}| + 1$. We denote the space of all linear orders over C by $\mathcal{L}(C)$. We denote by $(\mathcal{R}_{-i}, L)$ the preference profile obtained from $\mathcal{R}$ by replacing R_i with L.

A *voting correspondence* $\mathcal{F}$ is a mapping that, given a candidate set C and a preference profile $\mathcal{R}$ over C outputs a non-empty subset of candidates $S \subseteq C$; we write $S = \mathcal{F}(\mathcal{R})$. The candidates in S are called the *winners* of election $(C, \mathcal{R})$. A *voting correspondence* $\mathcal{F}$ is said to be a *voting rule* if it always produces a unique winner, i.e., $|\mathcal{F}(\mathcal{R})| = 1$ for any profile $\mathcal{R}$.

A voting correspondence can be transformed into a voting rule with the help of a *tie-breaking rule*. A tie-breaking rule for an election $(C, \mathcal{R})$ is a mapping $T = T(\mathcal{R}, S)$ that for any $S \subseteq C$, $S \neq \emptyset$, outputs a candidate $c \in S$. A tie-breaking rule T is *lexicographic* with respect to a preference ordering $\succ$ over C if for any preference profile $\mathcal{R}$ over C and any $S \subseteq C$ it selects the most preferred candidate from S with respect to $\succ$, i.e., we have $T(S) = c$ if and only if $c \succ a$ for all $a \in S \setminus \{c\}$. In the context of single-voter manipulation problems, where there is one voter that considers lying about his vote to obtain a better outcome, of particular interest are *benevolent* and *adversarial* tie-breaking rules: the former breaks ties in the manipulator's favor while the latter breaks ties against the manipulator's wishes (i.e., tie-breaking is lexicographic with respect to, respectively, the manipulator's true preference ordering and its inverse). In the traditional computational social choice terminology benevolent and adversarial tie-breaking correspond to, respectively, non-unique and unique winner settings.

Voting rules We will now describe the voting correspondences considered in this paper. All these correspondences assign scores to candidates; the winners are the candidates with the highest scores. In what follows, we will assume that these correspondences are transformed into voting rules by breaking ties adversarially; however, all of our results can be adapted in a straightforward manner to benevolent or, more generally, lexicographic tie-breaking.

Scoring rules Any vector $\alpha = (\alpha_1, \dots, \alpha_m) \in \mathbb{R}^m$ such that $\alpha_1 \geq \dots \geq \alpha_m$ defines a *scoring rule* $\mathcal{F}_\alpha$ as follows. Each voter grants α_i points to the candidate she ranks in the i-th position; the score of a candidate is the sum of the scores he receives from all voters. The vector α is called a *scoring vector*; we assume without loss of generality that the coordinates of α are nonnegative integers given in binary. We remark that scoring rules are defined for a fixed number of candidates, and therefore do not quite fit our definition of a voting rule. Thus, one needs to consider *families* of scoring rules (one for every possible number of candidates). From the algorithmic perspective, it is natural to restrict our attention to *polynomial-time computable families of scoring rules*, where the scoring vector α^m for an m-candidate election can be computed in time $\mathrm{poly}(m)$. Two well-known examples of such families are *Borda*, given by $\alpha = (m-1, \dots, 1, 0)$, and *k-approval*, given by $\alpha_i = 1$ if $i \leq k$, $\alpha_i = 0$ if $i > k$.

Bucklin Given an n-voter election, the *Bucklin winning round* is the smallest value of r such that the r-approval score of at least one candidate exceeds $n/2$. The *Bucklin score* of a candidate $c \in C$ is his r-approval score, where r is the Bucklin winning round.

Copeland A candidate a is said to win a *pairwise election* against b if more than half of the voters prefer a to b; if exactly half of the voters prefer a to b, then a is said to *tie* his pairwise election against b. Under the Copeland$^\alpha$ rule, $\alpha \in \mathbb{Q} \cap [0, 1]$, each candidate gets 1 point for each pairwise election he wins and α points for each pairwise election he ties.

Maximin The *Maximin score* of a candidate $c \in C$ is given by the number of votes c gets in his worst pairwise election, i.e., $\min_{d \in C \setminus \{c\}} |\{i \mid c \succ_i d\}|$.

Distances A *distance* on a space X is a mapping $d : X \times X \to \mathbb{R}$ that has the following properties for all $x, y, z \in X$: (1) non-negativity: $d(x, y) \geq 0$; (2) identity of indiscernibles: $d(x, y) = 0$ if and only if $x = y$; (3) symmetry: $d(x, y) = d(y, x)$; (4) triangle inequality: $d(x, y) + d(y, z) \geq d(x, z)$.

In this paper, we will be interested in distances over votes, i.e., mapping of the form $d : \mathcal{L}(C) \times \mathcal{L}(C) \to \mathbb{R}$. In fact, since we are interested in asymptotic complexity results, we will consider *families of distances* $(d^m)_{m \geq 1}$, where d^m is a distance over the space of all linear orderings of the set $\{c_1, \dots, c_m\}$. Specifically, we will consider three such families (in the following definitions, $C = \{c_1, \dots, c_m\}$ and R and L are two preference orders in $\mathcal{L}(C)$, also denoted as $\succ_R$ and $\succ_L$):

Swap distance. The *swap distance* $\mathrm{d_{swap}}(L, R)$ is given by

$$\mathrm{d_{swap}}(L, R) = |\{(c_i, c_j) \mid c_i \succ_L c_j \text{ and } c_j \succ_R c_i\}|.$$

This distance counts the number of swaps of adjacent candidates needed to transform L into R.

Footrule distance. The *footrule distance* $\mathrm{d_{fr}}(L,R)$ is given by

$$\mathrm{d_{fr}}(L,R)=\sum_{i=1}^{m}|r(c_i,L)-r(c_i,R)|.$$

This distance calculates by how much each candidate needs to be shifted to transform L into R, and sums up all shifts.

Maximum displacement distance. The *maximum displacement distance* $\mathrm{d_{md}}(L,R)$ is given by

$$\mathrm{d_{md}}(L,R)=\max_{i=1,\dots,m}|r(c_i,L)-r(c_i,R)|.$$

This distance is similar to the footrule distance; the only difference is that instead of summing up all shifts it only considers the maximum shift.

It is not hard to verify that the swap distance, the footrule distance, and the maximum displacement distance fulfill all distance axioms. It is also known [5] that the swap distance and the footrule distance are always within a factor of two from each other: we have $\mathrm{d_{swap}}(L,R)\le \mathrm{d_{fr}}(L,R)\le 2\mathrm{d_{swap}}(L,R)$ for any space of candidates C and any $L,R\in\mathcal{L}(C)$.

3. OUR MODEL

We will now formally describe our computational problem.

DEFINITION 3.1. *Let $\mathcal{D}=(d^m)_{m\ge 1}$ be a family of integer-valued distances, where d^m is a distance over $\mathcal{L}(\{c_1,\dots,c_m\})$. Let $\mathcal{F}$ be a voting rule. An instance of $(\mathcal{D},\mathcal{F})$-OPTMANIPULATION is given by an election $(C,\mathcal{R})$ with $C=\{c_1,\dots,c_m\}$, $\mathcal{R}=(R_1,\dots,R_n)$, a voter $i\in\{1,\dots,n\}$, a candidate $p\in C$, and a positive integer k. It is a "yes"-instance if there exists a vote $L\in\mathcal{L}(C)$ such that $\mathcal{F}(C,(\mathcal{R}_{-i},L))=\{p\}$ and $d^m(R_i,L)\le k$, and a "no"-instance otherwise.*

REMARK 3.2. The problem $(\mathcal{D},\mathcal{F})$-OPTMANIPULATION is in NP as long as all distances in $\mathcal{D}$ and the rule $\mathcal{F}$ are poly-time computable: one can guess a vote L and check that $\mathcal{F}(C,(\mathcal{R}_{-i},L))=\{p\}$ and $d^m(R_i,L)\le k$. In particular, it is in NP for all distance families and voting rules considered in this paper.

REMARK 3.3. We formulated OPTMANIPULATION as a decision problem. However, it also admits a natural interpretation as an optimization problem: in this case, we are given an election $(C,\mathcal{R})$, a voter i and a candidate p, and the goal is to find the smallest value of k such that there exists a vote $L\in\mathcal{L}(C)$ at distance at most k from R_i that satisfies $\mathcal{F}(C,(\mathcal{R}_{-i},L))=\{p\}$ (k is assumed to be $+\infty$ if there is no vote L with $\mathcal{F}(C,(\mathcal{R}_{-i},L))=\{p\}$). In this version of the problem, one can relax the optimality condition, and ask for an *approximately optimal* manipulative vote: an algorithm is said to be a *ρ-approximation algorithm* for $(\mathcal{D},\mathcal{F})$-OPTMANIPULATION, $\rho\ge 1$, if, given an instance of the problem for which the correct answer is $k\in\mathbb{R}\cup\{+\infty\}$, it outputs a value k' that satisfies $k\le k'\le\rho k$. We will consider the optimization version of OPTMANIPULATION (and prove hardness of approximation results) for Copeland and Maximin under swap distance (Sections 4) and footrule distance (Section 5).

REMARK 3.4. In our definition of OPTMANIPULATION, the manipulator wants to make a specific candidate elected; the identity of this candidate is given as a part of the instance description. An alternative approach would be to ask if the manipulator can obtain what he considers a better outcome by submitting a non-truthful vote, i.e., whether there is a vote $L\in\mathcal{L}(C)$ such that $d^m(R_i,L)\le k$ and $\mathcal{F}(C,(\mathcal{R}_{-i},L))\succ_i\mathcal{F}(C,\mathcal{R})$; we will refer to this problem as OPTMANIPULATION$'$. Clearly, an efficient algorithm for OPTMANIPULATION can be used to solve OPTMANIPULATION$'$, by determining the winner w under truthful voting, and then running the OPTMANIPULATION algorithm for all candidates that the manipulator ranks above w. Hence, OPTMANIPULATION is at least as hard as OPTMANIPULATION$'$. In what follows, we will provide polynomial-time algorithms for the "harder" problem OPTMANIPULATION. On the other hand, all our NP-hardness results apply to the "easier" problem OPTMANIPULATION$'$: in fact, in all our hardness proofs the manipulator's goal will be to make his favorite candidate the election winner. Using OPTMANIPULATION as our base problem allows for a direct comparison between the problem of finding the optimal manipulation and the swap bribery problem (see Section 7).

4. SWAP DISTANCE

We start by considering optimal manipulability with respect to what is perhaps the best known distance on votes, namely, the swap distance $\mathrm{d_{swap}}$.

4.1 Scoring Rules and Bucklin

The main result of this section is a simple polynomial-time algorithm that solves OPTMANIPULATION for swap distance and an arbitrary scoring rule; we then show that this algorithm can be adapted to work for the Bucklin rule.

An observation that will be important for our analysis of scoring rules in this and subsequent sections is that once we select the position of the manipulator's preferred candidate p, we know his final score. Thus, once p's position is fixed, it remains to rank other candidates so that their scores remain strictly lower than that of p (recall that we use adversarial tie-breaking). More formally, let $s_\alpha(c)$ be the total number of points a candidate c receives from non-manipulators under a voting rule $\mathcal{F}_\alpha$; we will say that a position j is *safe* for a candidate c_ℓ given that p is ranked in position f if $s_\alpha(c_\ell)+\alpha_j<s_\alpha(p)+\alpha_f$. Clearly, for a manipulation to be successful, all candidates other than p should be ranked in positions that are safe for them.

Fix a scoring rule $\mathcal{F}_\alpha$ with $\alpha=(\alpha_1,\dots,\alpha_m)$. Our algorithm relies on a subroutine $\mathcal{A}$ that given an election $(C,\mathcal{R})$ with $|C|=m$, a voter i, a candidate p, and a position f in i's vote, finds an optimal manipulation for i among all votes that rank p in position f. More formally, let

$$\mathcal{L}^f(\alpha)=\{L\in\mathcal{L}(C)\mid\mathcal{F}_\alpha(C,(\mathcal{R}_{-i},L))=\{p\},r(p,L)=f\};$$

our subroutine outputs $\perp$ if $\mathcal{L}^f(\alpha)$ is empty and a vote $\hat{L}$ such that $\mathrm{d_{swap}}(\hat{L},R_i)\le\mathrm{d_{swap}}(L,R_i)$ for all $L\in\mathcal{L}^f(\alpha)$ otherwise. Given $\mathcal{A}$, we can easily solve $(\mathrm{d_{swap}},\mathcal{F}_\alpha)$-OPTMANIPULATION: we run $\mathcal{A}$ for all values of f between 1 and m and output "yes" if at least one of these calls returns a vote $\hat{L}$ with $\mathrm{d_{swap}}(\hat{L},R_i)\le k$. Thus the running time of our algorithm is m times the running time of $\mathcal{A}$. It remains to describe $\mathcal{A}$.

THEOREM 4.1. *For any $\alpha=(\alpha_1,\dots,\alpha_m)\in{\mathbb{Z}_+}^m$ there exists a procedure $\mathcal{A}$ that takes an n-voter m-candidate election $(C,\mathcal{R})$, a voter $i\in\{1,\dots,n\}$, a candidate $p\in C$, and a position $f\in\{1,\dots,m\}$ as its input, outputs $\perp$ if $\mathcal{L}^f(\alpha)=\emptyset$ and a vote $\hat{L}$ that satisfies $\mathrm{d_{swap}}(\hat{L},R_i)\le\mathrm{d_{swap}}(L,R_i)$ for all $L\in\mathcal{L}^f(\alpha)$ otherwise, and runs in time $O(m^2\log(n\alpha_1))$.*

PROOF. For convenience, let us renumber the candidates in C so that $c_m=p$ and $c_1\succ_i\dots\succ_i c_{m-1}$. Our algorithm proceeds in

$m-1$ rounds. In the ℓ-th round, $\ell = 1, \dots, m-1$, we determine the final position of candidate c_ℓ; we then say that this candidate is *pinned* to that position, and the position becomes *unavailable*. Initially, all candidates are unpinned and all positions are available.

Initialization: We pin p to position f (thus f becomes unavailable), and then fill the remaining positions with the candidates in $C \setminus \{p\}$, in the order of i's preferences, i.e., placing c_1 in the highest available position and c_{m-1} in the lowest available position. In what follows, we will shift the candidates around in order to make p the winner.

Round ℓ, $\ell = 1, \dots, m-1$ Suppose that in the beginning of the round candidate c_ℓ is ranked in position j. If j is safe for c_ℓ, we pin c_ℓ to position j (which then becomes unavailable) and proceed to the next round. Otherwise, we find the smallest value of h such that position h is available and safe for c_ℓ; if no such value of h can be found, we terminate and return $\perp$. If a suitable value of h has been identified (note that $h > j$), then c_ℓ gets pinned to position h, and all unpinned candidates in positions $j+1, \dots, h$ are shifted one available position upwards.

If $\mathcal{A}$ does not abort (i.e., return $\perp$), it terminates at the end of the $(m-1)$-st round and returns the vote obtained at that point. Each round involves $O(m)$ score comparisons and shifts, and each comparison can be performed in time $O(\log(n\alpha_1))$; this implies the bound of $O(m^2 \log(n\alpha_1))$ on the running time. It remains to argue that $\mathcal{A}$ works correctly.

The following observation will be useful for our analysis.

LEMMA 4.2. *Suppose that at the beginning of round ℓ candidate c_ℓ is ranked in position j. Then positions $1, \dots, j-1$ are not available at that point.*

PROOF. An easy inductive argument shows that the set of candidates ranked above c_ℓ at the beginning of round ℓ is a subset of $\{c_1, \dots, c_{\ell-1}\}$. For each $t = 1, \dots, \ell-1$, candidate c_t is pinned in round t and therefore by the beginning of round ℓ his position is unavailable. As this holds for all positions above j, the lemma is proved. $\square$

We split the rest of proof into two lemmas.

LEMMA 4.3. *If the subroutine $\mathcal{A}(C, \mathcal{R}, i, p, f)$ outputs a vote $\hat{L}$ then $\hat{L} \in \mathcal{L}^f(\alpha)$, and if it outputs $\perp$ then $\mathcal{L}^f(\alpha) = \emptyset$.*

PROOF. By construction, if $\mathcal{A}$ outputs a vote $\hat{L}$, then $r(p, \hat{L}) = f$. Moreover, every other candidate c_j can only be pinned to a position that is safe for him. Since $\mathcal{A}$ returns $\hat{L}$ only when all candidates in C are pinned, we have $\mathcal{F}_\alpha(C, (\mathcal{R}_{-i}, \hat{L})) = \{p\}$, and hence $\hat{L} \in \mathcal{L}^f(\alpha)$.

Now, suppose that $\mathcal{A}(C, \mathcal{R}, i, p, f) = \perp$. This means that for some candidate c_ℓ, $\ell \le m-1$, our algorithm was unable to find an available safe position. Let $\hat{L}$ be the vote constructed by the algorithm by the beginning of round ℓ, and let h be the lowest available position at the beginning of round ℓ.

Suppose for the sake of contradiction that $\mathcal{L}^f(\alpha) \neq \emptyset$, and let L be some vote in $\mathcal{L}^f(\alpha)$. Since the algorithm has output $\perp$, position h is not safe for c_ℓ. Thus, in L candidate c_ℓ is ranked in position $h+1$ or lower. Consequently, some candidate c_t that is ranked in position $h+1$ or lower in $\hat{L}$ must be ranked in position h or higher in L. Since positions $h+1, \dots, m$ are not available at the beginning of round ℓ, they are occupied by candidates who were pinned to these positions in earlier rounds (and, possibly, by p), i.e., $t < \ell$. This means that position h was available when c_t was processed, but the algorithm chose not to place c_t in position h. By Lemma 4.2, it was not the case that c_t was pinned to the position it was in at the beginning of round t. Hence, the reason why c_t was ranked in position $h+1$ or lower was that h (and, *a forteriori*, any position above h) was not safe for c_t. On the other hand, we have argued that c_t is ranked in position h or higher in L, a contradiction with $L \in \mathcal{L}^f(\alpha)$. Thus it has to be the case that $\mathcal{L}^f(\alpha) = \emptyset$. $\square$

LEMMA 4.4. *If $\mathcal{A}(C, \mathcal{R}, i, p, f) = \hat{L}$, then $\mathrm{d}_{\mathrm{swap}}(\hat{L}, R_i) \le \mathrm{d}_{\mathrm{swap}}(L, R_i)$ for all $L \in \mathcal{L}^f(\alpha)$.*

PROOF. We will prove a somewhat stronger statement: there is a unique optimal vote in $\mathcal{L}^f(\alpha)$, and this vote coincides with $\hat{L}$. Suppose for the sake of contradiction that there exists a vote $L \in \mathcal{L}^f(\alpha)$ such that $\mathrm{d}_{\mathrm{swap}}(L, R_i) \le \mathrm{d}_{\mathrm{swap}}(L', R_i)$ for all $L' \in \mathcal{L}^f(\alpha)$ and $L \neq \hat{L}$. Let c_ℓ be the first candidate ranked differently by L and $\hat{L}$, i.e., $\ell = \min\{j \mid r(c_j, L) \neq r(c_j, \hat{L})\}$.

Suppose first that $r(c_\ell, \hat{L}) > r(c_\ell, L)$. It cannot be the case that c_ℓ remains in place during round ℓ: by Lemma 4.2 all positions above c_ℓ in $\hat{L}$ are filled with candidates in $\{c_1, \dots, c_{\ell-1}\}$, and $r(c_j, \hat{L}) = r(c_j, L)$ for $j < \ell$. Hence, c_ℓ has to move during round ℓ. Now, $r(c_\ell, \hat{L})$ is the highest available position that is safe for c_ℓ. Since $r(c_\ell, L)$ is necessarily safe, it follows that $r(c_\ell, L)$ must be unavailable at the beginning of round ℓ. However, this means that there is a candidate c_j, $j < \ell$, pinned to this position in $\hat{L}$, and all such candidates are ranked in the same positions in L and $\hat{L}$, a contradiction.

Thus, it has to be the case that $r(c_\ell, \hat{L}) < r(c_\ell, L)$. Let c_j be the candidate ranked in position $r(c_\ell, \hat{L})$ in L; we have $j > \ell$ by our choice of ℓ. Let L' be the vote obtained from L by swapping c_ℓ and c_j. We claim that $L' \in \mathcal{L}^f(\alpha)$ and $\mathrm{d}_{\mathrm{swap}}(L', R_i) < \mathrm{d}_{\mathrm{swap}}(L, R_i)$, thus contradicting our choice of L.

To see that $L' \in \mathcal{L}^f(\alpha)$, observe that after the swap the scores of all candidates other than c_ℓ do not go up and $r(c_\ell, L') = r(c_j, L) = r(c_\ell, \hat{L})$, so position $r(c_\ell, L')$ is safe for c_ℓ. It remains to prove that $\mathrm{d}_{\mathrm{swap}}(L', R_i) < \mathrm{d}_{\mathrm{swap}}(L, R_i)$. To this end, we need and additional definition: we say that a pair of candidates (c, c') is an *inversion* in a vote R if $r(c, R_i) < r(c', R_i)$, but $r(c, R) > r(c', R)$. Clearly, the swap distance from R to R_i is simply the number of inversions in R. Thus, our goal is to show that L' has fewer inversions than L.

Observe first that (c_j, c_ℓ) is an inversion in L, but not in L'. Among all other pairs of candidates, it suffices to consider pairs of the form (c_j, c) and (c, c_ℓ), where c is ranked between c_j and c_ℓ in L; any other pair of candidates is an inversion in L if and only if it is an inversion in L'.

Since $j > \ell$, we have three possibilities:

$c_\ell \succ_i c \succ_i c_j$. In this case, both (c_j, c) and (c, c_ℓ) are inversions in L, but neither of them is an inversion in L'.

$c_\ell \succ_i c_j \succ_i c$. In this case, (c, c_ℓ) is an inversion in L, but (c_j, c) is not. On the other hand, (c, c_j) is an inversion in L', but (c_ℓ, c) is not.

$c \succ_i c_\ell \succ_i c_j$. In this case, (c_j, c) is an inversion in L, but (c, c_ℓ) is not. On the other hand, (c_ℓ, c) is an inversion in L', but (c, c_j) is not.

Thus, for any candidate c ranked between c_j and c_ℓ in L the pairs involving c contribute at least as much to the inversion count of L as to that of L'. By taking into account the pair (c_j, c_ℓ) itself, we conclude that $\mathrm{d}_{\mathrm{swap}}(L', R_i) < \mathrm{d}_{\mathrm{swap}}(L, R_i)$, a contradiction.

It follows that $\hat{L}$ is the optimal vote in $\mathcal{L}^f(\alpha)$ and the proof of the lemma is complete. $\square$

The theorem now follows easily from Lemmas 4.3 and 4.4. □

We have already explained how to convert the subroutine $\mathcal{A}$ into an algorithm for OPTMANIPULATION. Thus, we obtain the following corollary.

COROLLARY 4.5. *For every polynomial-time computable family $\hat{\mathcal{F}} = (\mathcal{F}_\alpha^m)_{m=1,\dots}$ of scoring rules, the problem $(\mathrm{d}_{\mathrm{swap}}, \hat{\mathcal{F}})$-*OPTMANIPULATION *is in* P.

For the Bucklin rule, the algorithm is essentially the same; the only difference is in the definition of a safe position.

THEOREM 4.6. $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Bucklin})$-OPTMANIPULATION *is in* P.

PROOF SKETCH. Consider an election $(C, \mathcal{R})$ and a manipulator i. Just as in the proof of Theorem 4.1, it suffices to design a procedure that, for a given value of $f \in \{1, \dots, m\}$, searches for the best manipulative vote that ranks p in position f and returns $\perp$ if no such vote can make p the unique winner.

Fix a particular value of f, and let $\mathcal{L}_f = \{L \in \mathcal{L}(C) \mid r(p, L) = f\}$. Let L_f be an arbitrary vote in $\mathcal{L}_f$. Let r^* be the smallest value of r such that p's r-approval score in $(C, (\mathcal{R}_{-i}, L_f))$ is greater than $n/2$; note that r^* does not depend on the choice of L_f. For every candidate $c \in C$, and every $r = 1, \dots, m$, let $s_r(c)$ denote c's r-approval score in $(C, \mathcal{R}_{-i})$, and let s be p's r^*-approval score in $(C, (\mathcal{R}_{-i}, L_f))$; note that $s > n/2$.

To make p the winner, we need to ensure that r^* is the Bucklin winning round and that the r^*-approval score of any candidate $c \in C \setminus \{p\}$ does not exceed s. Thus, if there is a candidate $c \in C \setminus \{p\}$ such that $s_r(c) > n/2$ for some $r < r^*$ or $s_{r^*}(c) \geq s$, then there is no vote in $\mathcal{L}_f$ that makes p the unique election winner, so we return $\perp$ and stop.

Now, suppose that this is not the case. Set $C_1 = \{c \in C \setminus \{p\} \mid s_{r^*}(c) = s - 1\}$, $C_2 = \{c \in C \setminus (C_1 \cup \{p\}) \mid s_r(c) = \lfloor \frac{n}{2} \rfloor$ for some $r < r^*\}$. Intuitively, candidates from C_1 can prevent p from winning by receiving the same r^*-approval score as p, which happens if they are ranked in the top r^* positions. Similarly, candidates from C_2 can prevent p from winning by receiving a strict majority vote in an earlier round; this happens if they are ranked in the top $r^* - 1$ positions. Thus, p is the unique Bucklin winner in the election where the manipulator submits a vote $L \in \mathcal{L}_f$ if and only if (a) $r(c, L) > r^*$ for all $c \in C_1$ and (b) $r(c, L) \geq r^*$ for all $c \in C_2$. We will say that a position j is *safe* for a candidate $c \in C \setminus \{p\}$ if (1) $c \notin C_1 \cup C_2$ or (2) $c \in C_1$ and $j > r^*$ or (3) $c \in C_2$ and $j \geq r^*$. The argument above shows that p is the unique Bucklin winner in $(C, (\mathcal{R}_{-i}, L))$ if and only if in L each candidate $c \neq p$ is ranked in a position that is safe for him.

Given this definition of a safe position, we can apply the algorithm for scoring rules described in the proof of Theorem 4.1; note that this algorithm operates in terms of safe positions rather than actual scores. The proofs of correctness and optimality are identical to those for scoring rules (these proofs, too, are phrased in terms of safe positions). □

4.2 Maximin and Copeland

For both Maximin and Copeland, finding an optimal manipulation with respect to the swap distance turns out to be computationally hard. In fact, we will prove that the optimization versions of these problems (see Remark 3.3) cannot be approximated up to a factor of $\delta \log |C|$ for some $\delta > 0$ unless P=NP; this implies, in particular, that the decision versions of these problems are NP-hard (and hence, by Remark 3.2, NP-complete).

We provide reductions from the optimization version of the SET COVER problem [10]. Recall that an instance of SET COVER is given by a ground set $G = \{g_1, \dots, g_t\}$ and a collection $\mathcal{S} = \{S_1, \dots, S_r\}$ of subsets of G. In the optimization version of the problem, the goal is to find the smallest value of h such that G can be covered by h sets from $\mathcal{S}$; we denote this value of h by $h(G, \mathcal{S})$. More formally, we are interested in the smallest value of h such that $G = \cup_{S' \in \mathcal{S}'} S'$ for some collection of subsets $\mathcal{S}' \subseteq \mathcal{S}$ with $|\mathcal{S}'| = h$. A *ρ-approximation algorithm* for SET COVER is a procedure that, given an instance $(G, \mathcal{S})$ of set cover, outputs a value h' that satisfies $h(G, \mathcal{S}) \leq h' \leq \rho \cdot h(G, \mathcal{S})$. There exists a $\delta > 0$ such that SET COVER does not admit a polynomial-time $\delta \log t$-approximation algorithm unless P=NP [17]. The inapproximability result still holds if we assume that (1) $G = \cup_{S \in \mathcal{S}} S$; (2) $t \leq r$; and (3) $r \leq t^K$ for some positive constant K. Indeed, if (1) fails, the instance does not admit a solution, (2) can be achieved by duplicating sets in $\mathcal{S}$, and (3) follows by a careful inspection of the proof in [17]. Thus, in what follows, we only consider instances of SET COVER that satisfy conditions (1)–(3).

THEOREM 4.7. *There exists a $\delta > 0$ s. t.* $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Maximin})$-OPTMANIPULATION *does not admit a polynomial-time $\delta \log |C|$-approximation algorithm unless* P=NP.

PROOF. Suppose that we are given an instance $(G, \mathcal{S})$ of SET COVER with $G = \{g_1, \dots, g_t\}$, $\mathcal{S} = \{S_1, \dots, S_r\}$ that satisfies conditions (1)–(3).

In our election, the candidate set is $C = \{p\} \cup G \cup X \cup S$, where $X = \{x_1, \dots, x_{2r}\}$ and $S = \{s_1, \dots, s_r\}$.

The proof of McGarvey theorem [14] implies that we can construct a preference profile $\mathcal{R}'$ with n' voters, where n' is polynomially bounded in t and r, so that n' is even and:

- For any $c \in C \setminus \{p\}$ exactly $n'/2 - 2$ voters prefer p to c.
- For any $S_j \in \mathcal{S}$ and any $g_\ell \in S_j$ exactly $n'/2 - 2$ voters prefer g_ℓ to s_j.
- For any other pair of candidates $(c, c') \in G \cup S \times G \cup S$, exactly $n'/2$ voters prefer c to c'.
- For $j = 1, \dots, 2r - 1$ exactly $n'/2 - 4$ voters prefer x_j to x_{j+1}, and $n'/2 - 4$ voters prefer x_{2r} to x_1.
- For any $g_j \in G$ and any $x \in X$ exactly $n'/2$ voters prefer g_j to x.
- For any $s_j \in S$ and any $x \in X$ exactly $n'/2 - 4$ voters prefer s_j to x.

Denote the Maximin score of candidate c in election $(C, \mathcal{R}')$ by $s(c)$. We have $s(p) = n'/2 - 2$, $s(g_j) = n'/2 - 2$ for any $g_j \in G$ (this follows from condition (1)), $s(s_j) = n'/2 - 4$ for any $s_j \in S$, and $s(x_j) = n'/2 - 4$ for any $x_j \in X$.

We let $n = n' + 1$, $i = n$ and set our preference profile to be $\mathcal{R} = (\mathcal{R}', R_n)$, where voter n (the manipulator) ranks the candidates as

$$p \succ g_1 \succ \dots \succ g_t \succ x_1 \succ \dots \succ x_{2r} \succ s_1 \succ \dots \succ s_r.$$

This completes the description of our $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Maximin})$-OPTMANIPULATION instance (as we consider the optimization version of the problem, we need not specify k).

Observe that p's final Maximin score is $n'/2 - 1$ if and only if the manipulator ranks p first. Further, the final Maximin score of any candidate in $X \cup S$ is at most $n'/2 - 3$. Finally, the final Maximin score of a candidate $g_j \in G$ is $n'/2 - 1$ if in the manipulator's vote g_j appears above all candidates s_ℓ such that $g_j \in S_\ell$ and $n'/2 - 2$ otherwise. Thus, to make p the unique winner, the manipulator should rank him first, and rank each candidate $g_j \in G$ below a candidate representing a set that covers g_j.

Suppose that $h(G, \mathcal{S}) = h$, i.e., there exists a collection of subsets $\mathcal{S}' = \{S_{i_1}, \dots, S_{i_h}\}$ with $i_1 < \dots < i_h$ such that $\cup_{S' \in \mathcal{S}'} S' =$

G. Consider a vote L that ranks p first, followed by candidates $s_{i_1}, \dots, s_{i_h}$ (in this order), followed by candidates in $X \cup G$ (in the order of their appearance in R_n), followed by the remaining candidates in S (in the order of their appearance in R_n). By the argument above, p is the unique Maximin winner of $(C, (\mathcal{R}', L))$. Furthermore, we have $\mathrm{d}_{\mathrm{swap}}(L, R_n) \le h(t + 2r + (r - h))$: to transform R_n into L, we swap each of the candidates s_{i_j}, $j = 1, \dots, h$, with (a) t candidates in G, (b) $2r$ candidates in X and (c) at most $r - h$ candidates in S. By condition (2), we obtain $\mathrm{d}_{\mathrm{swap}}(L, R_n) \le 4hr$.

On the other hand, consider an arbitrary vote L' such that p is the unique Maximin winner of $(C, (\mathcal{R}', L'))$. Construct a bipartite graph with the vertex set $G \cup S$ in which there is an edge between g_j and s_ℓ if and only if s_ℓ is ranked above g_j in L'. We claim that this graph contains a matching of size h. To see this, consider a greedy algorithm that constructs a matching by inspecting the vertices in G one by one and matching each vertex to one of its previously unmatched neighbors in S; if some vertex in G cannot be matched, the algorithm proceeds to the next vertex. If this algorithm terminates without finding h edges, it means that the matched vertices in S correspond to a cover of size at most $h - 1$, a contradiction with $h(G, \mathcal{S}) = h$.

Consider a pair of candidates (g_j, s_ℓ) that corresponds to an edge of this matching, and an arbitrary candidate $x \in X$. It cannot be the case that L' ranks g_j above x and x above s_ℓ: otherwise, by transitivity, L' would rank g_j above s_ℓ. Therefore, at least one of the pairs (g_j, x) and (x, s_ℓ) is ordered differently in R_n and L', and therefore each edge of the matching contributes at least $2r$ to the swap distance between L' and R_n. Summing over all edges of the matching, we obtain that $\mathrm{d}_{\mathrm{swap}}(R_n, L') \ge 2hr$.

Now, suppose that there is a polynomial-time ρ-approximation algorithm $\mathcal{M}$ for $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Maximin})$-OPTMANIPULATION: given an instance of $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Maximin})$-OPTMANIPULATION that admits a successful manipulative vote L with $\mathrm{d}_{\mathrm{swap}}(L, R_i) = k$, this algorithm outputs a value k' that satisfies $k \le k' \le \rho k$. Consider the following algorithm $\mathcal{M}'$ for SET COVER: given an instance $(G, \mathcal{S})$ of SET COVER with $|G| = t$, $|\mathcal{S}| = r$, $\mathcal{M}'$ transforms it into an instance of $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Maximin})$-OPTMANIPULATION as described above, applies $\mathcal{M}$, and divides the returned value by $2r$. Clearly, $\mathcal{M}'$ runs in polynomial time. We claim that it provides a 2ρ-approximation algorithm for SET COVER.

Indeed, let $h = h(G, \mathcal{S})$. Then for the corresponding instance of $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Maximin})$-OPTMANIPULATION there exists a successful manipulative vote L with $\mathrm{d}_{\mathrm{swap}}(L, R_i) \le 4hr$ and hence $\mathcal{M}$ outputs a value k' that satisfies $k' \le 4\rho hr$. On the other hand, for any successful manipulative vote L' we have $\mathrm{d}_{\mathrm{swap}}(L', R_i) \ge 2hr$, and hence the value k' output by $\mathcal{M}$ satisfies $k' \ge 2hr$. Thus, $\mathcal{M}$ produces a value h' that satisfies $h \le h' \le 2\rho h$.

Since $|C| = O(t + r)$ and, by condition (3), $r \le t^K$ (where K is a constant whose value can be extracted from the proof in [17]), we have $\log |C| \le \gamma \log t$ for a suitable constant $\gamma > 0$. Therefore, if there exists a polynomial-time $(\delta' \log |C|)$-approximation algorithm for $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Maximin})$-OPTMANIPULATION for $\delta' > 0$, then there exists a polynomial-time $(2\delta'\gamma \log t)$-approximation algorithm for SET COVER. By [17], for small enough δ this implies P=NP. □

The argument for Copeland is similar.

THEOREM 4.8. *There exists a $\delta > 0$ such that for any $\alpha \in \mathbb{Q} \cap [0, 1]$, $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Copeland}^\alpha)$-OPTMANIPULATION does not admit a polynomial-time $\delta \log |C|$-approximation algorithm unless* P=NP.

PROOF SKETCH. Suppose that we are given an instance $(G, \mathcal{S})$ of SET COVER with $G = \{g_1, \dots, g_t\}$, $\mathcal{S} = \{S_1, \dots, S_r\}$ that satisfies conditions (1)–(3); we will additionally assume that t and r are odd.

In our election, the candidate set is $C = \{p\} \cup G \cup X \cup S$, where $X = \{x_1, \dots, x_{6r}\}$ and $S = \{s_1, \dots, s_r\}$. There exists a tournament over the candidate set C such that that:

- p beats all candidates in $G \cup S$ as well as $6r - (t + 1)/2$ candidates in X, and loses to all other candidates in X.
- Every candidate $g_i \in G$ is tied with all candidates s_ℓ such that $g_i \in S_\ell$ and beats all other candidates in $X \cup S$.
- Every candidate in G beats exactly $(t-1)/2$ other candidates in G.
- Every candidate in $X \cup S$ beats exactly $(7r - 1)/2$ other candidates in $X \cup S$.

Thus, by McGarvey theorem [14], we can construct a preference profile $\mathcal{R}'$ with n' voters that generates this tournament; moreover, we can assume that n' is even and polynomially bounded in t and r, We let $n = n' + 1$, $i = n$ and set our preference profile to be $\mathcal{R} = (\mathcal{R}', R_n)$, where voter n (the manipulator) ranks the candidates as

$$p \succ g_1 \succ \dots \succ g_t \succ x_1 \succ \dots \succ x_{6r} \succ s_1 \succ \dots \succ s_r.$$

This completes the description of our $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Copeland}^\alpha)$-OPT-MANIPULATION instance; note that n is even and therefore the value of α is unimportant for our analysis.

Observe that in $\mathcal{R}$ the Copeland score of p is $(t - 1)/2 + 7r$, the Copeland score of each $g_j \in G$ is $(t - 1)/2 + 7r$, and the Copeland score of each candidate in $X \cup S$ is at most $(7r - 1)/2 + 1 < 4r$. Thus, under truthful voting p is not the unique winner; indeed, for p to be the unique winner, in the manipulator's vote every candidate $g_j \in G$ must be ranked below some candidate s_ℓ such that $g_j \in S_\ell$. Note also that the manipulator's vote can only affect the outcomes of pairwise elections for candidate pairs of the form (g_j, s_ℓ), $g_j \in S_\ell$. Thus, no matter how the manipulator votes, the Copeland score of every candidate $x \in X$ is at most $4r < (t - 1)/2 + 7r$, and the Copeland score of every candidate $s_\ell \in S$ is at most $4r + t < (t - 1)/2 + 7r$ (recall that we assume $t < r$), and hence candidates in $X \cup S$ are not among the election winners. We conclude that L is a successful manipulative vote if and only if it ranks each candidate $g_j \in G$ below a candidate representing a set that covers g_j, This condition is almost identical to the one in the proof of Theorem 4.7, and, from this point on, the proof repeats the proof of Theorem 4.7 almost verbatim; the reader can verify that the analysis is not negatively impacted by the fact that the set X contains $6r$ candidates (rather than $2r$ candidates, as in the proof of Theorem 4.7). □

5. FOOTRULE DISTANCE

For the footrule distance our analysis turns out to be much easier than for the swap distance: for scoring rules and Bucklin, we design a simple matching-based algorithm, and for Copeland and Maximin we can use the fact that the swap distance and the footrule distance are always within a factor of 2 from each other, as this allows us to inherit the hardness results of the previous section.

5.1 Scoring Rules and Bucklin

The overall structure of our argument is similar to the one in Section 4: for any scoring rule $\mathcal{F}_\alpha$ with $\alpha = (\alpha_1, \dots, \alpha_m)$ we will design a procedure $\mathcal{A}'$ that, given an election $(C, \mathcal{R})$ with $|C| = m$, a voter i, the preferred candidate p, a target position f for the preferred candidate, and a bound k on the distance, constructs a vote L such that (a) $\mathcal{F}(C, (\mathcal{R}_{-i}, L)) = \{p\}$; (b) $r(p, L) = f$; (c) $\mathrm{d}_{\mathrm{fr}}(L, R_i) \le k$, or returns $\perp$ if no such vote exists. We then run this procedure for $f = 1, \ldots, m$ and return "yes" if at least one of these calls *does not* return $\perp$.

We assume without loss of generality that the manipulator ranks the candidates as $c_1 \succ_i \ldots \succ_i c_m$ (note that this is different from the assumption we made in Section 4), and denote by $s_\alpha(c)$ the score of a candidate $c \in C$ in election $(C, \mathcal{R}_{-i})$ under the voting rule $\mathcal{F}_\alpha$. Let r be the rank of p in i's truthful vote, i.e., $p = c_r$.

$\mathcal{A}'$ proceeds by constructing a bipartite graph G with parts $X = C \setminus \{p\}$ and $Y = \{1, \ldots, m\} \setminus \{f\}$; there is an edge from c_j to ℓ if and only if position ℓ is safe for c_j, i.e., $s_\alpha(c_j) + \alpha_\ell < s_\alpha(p) + \alpha_f$, Each edge has a weight: the weight of the edge (c_j, ℓ) is simply $|j - \ell|$. Clearly, there is a one-to-one correspondence between votes L that rank p in position f and satisfy $\mathcal{F}_\alpha(C, (\mathcal{R}_i, L)) = \{p\}$ and perfect matchings in this graph. Furthermore, the cost of a matching M is x if and only if the corresponding vote L_M satisfies $\mathrm{d_{fr}}(L_M, R_i) = x + |r - f|$. Thus, it suffices to find a minimum cost perfect matching in G; our algorithm returns the vote L that corresponds to this matching if its cost does not exceed $k - |r - f|$ and $\perp$ otherwise. The graph G can be constructed in time $O(m^2 \log(n\alpha_1))$, and a minimum-cost matching can be found in time $O(m^3)$ [4].

We summarize these observations as follows.

THEOREM 5.1. *For every polynomial-time computable family $\hat{\mathcal{F}} = (\mathcal{F}_\alpha^m)_{m=1,\ldots}$ of scoring rules, the problem* $(\mathrm{d_{fr}}, \hat{\mathcal{F}})$-OPTMANIPULATION *is in* P.

For the Bucklin rule, it suffices to combine the matching-based algorithm given above with the definition of a safe position given in the proof of Theorem 4.6. We obtain the following corollary.

COROLLARY 5.2. $(\mathrm{d_{fr}}, \mathrm{Bucklin})$-OPTMANIPULATION *is in* P.

5.2 Maximin and Copeland

In Section 2 we have mentioned that for any candidate set C and any pair of votes $L, R \in \mathcal{L}(C)$ we have $\mathrm{d_{swap}}(L, R) \leq \mathrm{d_{fr}}(L, R) \leq 2\mathrm{d_{swap}}(L, R)$ [5].

Now, suppose that there exists a ρ-approximation algorithm $\mathcal{A}_{\mathrm{fr}}$ for $(\mathrm{d_{fr}}, \mathcal{F})$-OPTMANIPULATION for some voting rule $\mathcal{F}$. Consider an instance $(C, \mathcal{R}, i, p)$ of (the optimization version of) this problem, and let

$$\mathcal{L}' = \{L \in \mathcal{L}(C) \mid \mathcal{F}(C, (\mathcal{R}_{-i}, L)) = \{p\}\}.$$

If $\mathcal{L}' \neq \emptyset$, let $k = \min\{\mathrm{d_{fr}}(L, R_i) \mid L \in \mathcal{L}'\}$. On this instance $\mathcal{A}_{\mathrm{fr}}$ outputs a value k' that satisfies $k \leq k' \leq \rho k$; this value corresponds to a vote $L \in \mathcal{L}'$ such that $\mathrm{d_{fr}}(L, R_i) = k'$.

Now, for any vote $L' \in \mathcal{L}'$ we have

$$\mathrm{d_{swap}}(L', R_i) \geq \frac{1}{2}\mathrm{d_{fr}}(L', R_i) \geq \frac{k}{2}.$$

On the other hand, for L we obtain

$$\mathrm{d_{swap}}(L, R_i) \leq \mathrm{d_{fr}}(L, R_i) = k' \leq \rho k.$$

Now, consider an algorithm $\mathcal{A}_{\mathrm{swap}}$ for $(\mathrm{d_{swap}}, \mathcal{F})$-OPTMANIPULATION that, given an instance of the problem, runs $\mathcal{A}_{\mathrm{fr}}$ on it and returns the value reported by $\mathcal{A}_{\mathrm{fr}}$. The computation above proves that $\mathcal{A}_{\mathrm{swap}}$ is a 2ρ-approximation algorithm for $(\mathrm{d_{swap}}, \mathcal{F})$-OPTMANIPULATION (note that $\mathcal{A}_{\mathrm{swap}}$ returns $+\infty$ if and only if $\mathcal{L}' = \emptyset$). Combining this observation with Theorems 4.7 and 4.8, we obtain the following corollaries.

COROLLARY 5.3. *There exists a $\delta > 0$ s. t.* $(\mathrm{d_{fr}}, \mathrm{Maximin})$-OPTMANIPULATION *does not admit a poly-time $\delta \log |C|$-approximation algorithm unless* P=NP.

COROLLARY 5.4. *There exists a $\delta > 0$ such that for any $\alpha \in \mathbb{Q} \cap [0, 1]$,* $(\mathrm{d_{fr}}, \mathrm{Copeland}^\alpha)$-OPTMANIPULATION *does not admit a poly-time $\delta \log |C|$-approximation algorithm unless* P=NP.

6. MAX DISPLACEMENT DISTANCE

Maximum displacement distance is fairly generous to the manipulator. Indeed, the optimal manipulation problems for swap distance and footrule distance become trivial if the maximum distance k is bounded by a constant: in this case, there are only polynomially many possible manipulative votes, and the manipulator can try all of them. In contrast, for the maximum displacement distance, there are exponentially many votes even at distance 2 from the true vote (to see this, cut the manipulator's vote into segments of length 3; within each segment, the candidates can be shuffled independently). Nevertheless, from the algorithmic perspective maximum displacement distance exhibits essentially the same behavior as swap distance and footrule distance: we can design efficient algorithms for all scoring rules and the Bucklin rule, and derive NP-hardness results for Copeland and Maximin.

6.1 Scoring Rules and Bucklin

For scoring rules, we can use a simplified variant of the min-cost matching argument given in Section 5.1. Again, suppose that we are given a scoring rule $\mathcal{F}_\alpha$ with $\alpha = (\alpha_1, \ldots, \alpha_m)$, an election $(C, \mathcal{R})$ with $|C| = m$, a manipulator i, a preferred candidate p and a distance bound k. We assume that the manipulator ranks the candidates as $c_1 \succ_i \ldots \succ_i c_m$. For each $f = 1, \ldots, m$ we try to find a successful manipulative vote L with $\mathrm{d_{md}}(L, R_i) \leq k$ that ranks p in position f; in fact, it suffices to consider only values of f that satisfy $|f - r(p, R_i)| \leq k$. For each such f, we construct a bipartite graph G with parts $C \setminus \{p\}$ and $\{1, \ldots, m\} \setminus \{f\}$. In this graph, there is an edge from c_j to ℓ if and only if ℓ is safe for c_j (we use the same definition of a safe position as in Section 5.1) and $|\ell - j| \leq k$. In contrast to the construction in Section 5.1, the graph is unweighted. It is immediate that there is a one-to-one correspondence between perfect matchings in G and successful manipulative votes at distance at most k from R_i. Thus, we obtain the following result.

THEOREM 6.1. *For every polynomial-time computable family $\hat{\mathcal{F}} = (\mathcal{F}_\alpha^m)_{m=1,\ldots,}$ of scoring rules, the problem* $(\mathrm{d_{md}}, \hat{\mathcal{F}})$-OPTMANIPULATION *is in* P.

For the Bucklin rule, we use the same approach as in Section 5, i.e., combine the matching-based algorithm with the definition of a safe position given in the proof of Theorem 4.6. This results in following corollary.

COROLLARY 6.2. $(\mathrm{d_{md}},\mathrm{Bucklin})$-OPTMANIPULATION *is in* P.

6.2 Maximin and Copeland

For Maximin and Copeland, finding an optimal manipulation with respect to the maximum displacement distance is computationally hard; however, in contrast with our results in Sections 4 and 5, we are only able to show the NP-hardness of the decision version of this problem (rather than inapproximability of its optimization version). We omit the proofs of the following two theorems due to space constraints; both proofs are based on (somewhat involved) reductions from SET COVER.

THEOREM 6.3. $(\mathrm{d_{md}}, \mathrm{Maximin})$-OPTMANIPULATION *is* NP-*complete.*

THEOREM 6.4. *For any $\alpha \in \mathbb{Q} \cap [0, 1]$,* $(\mathrm{d_{md}}, \mathrm{Copeland}^\alpha)$-OPTMANIPULATION *is* NP-*complete.*

7. OPTIMAL MANIPULABILITY AND SWAP BRIBERY

The problem of finding an optimal manipulation with respect to the swap distance can be viewed as a special case of the swap bribery problem [8]. In the swap bribery model, there is an external party that wants to make a particular candidate the election winner. This party can pay the voters to change their preference orders, with a price assigned to swapping each pair of candidates in each vote. The goal is to decide whether the manipulator can achieve his goal given a budget constraint. Clearly, our problem is a special case of swap bribery, where for one voter each swap has unit cost, and for the remaining voters the prices are set to $+\infty$. Swap bribery is known to be hard, even to approximate, for almost all prominent voting rules, including such relatively simple rules as 2-approval. Thus, the easiness results of Section 4 identify a new family of easy instances of the swap bribery problem, thus complementing the results of [7, 6, 19]. It would be interesting to see if a somewhat more general variant of the swap bribery problem for scoring rules, where only one voter can be bribed but swap bribery prices can be arbitrary, remains tractable; it is not clear if the algorithm given in Section 4 can be adapted to handle this setting.

On the other hand, one may wonder if the hardness results of Section 4 are implied by the existing hardness results for swap bribery. However, this does not seem to be the case: the hardness (and inapproximability) of swap bribery for Copeland and Maximin follows from the hardness results for the possible winner problem [21], and the latter problem is easy if all but one voter's preferences are fixed (it can be verified that the algorithm of Bartholdi et al. [2] works even if the positions of some candidates in the vote are already fixed). Thus, the hardness results for Copeland and Maximin given in Section 4 strengthen the existing hardness results for swap bribery with respect to these rules.

8. CONCLUSIONS AND FUTURE WORK

We have considered the problem of finding a successful manipulative vote that differs from the manipulators' preferences as little as possible, for three distance measures on votes and four types of voting rules. Our results are summarized in Table 1 (where "NPC" stands for "NP-complete" and "$(\log m)$-inapp." stands for "inapproximable up to a factor of $\Omega(\log m)$").

A natural direction for future work is extending our results to other distances on votes; for instance, it should not be too hard to generalize our results for weighted variants of swap and footrule distances; such distances play an important role in several applications of rank aggregation, and have received considerable attention in the literature (see [13] and references therein). At a more technical level, we remark that for maximum displacement distance we only have NP-hardness results for Copeland and Maximin; it would be interesting to see if this variant of our problem admits efficient approximation algorithms.

	Sc. rules	Bucklin	Copeland	Maximin
d_{swap}	P	P	$(\log m)$-inapp.	$(\log m)$-inapp.
d_{fr}	P	P	$(\log m)$-inapp.	$(\log m)$-inapp.
d_{md}	P	P	NPC	NPC

Table 1: Summary of results

Acknowledgments This research was supported by National Research Foundation (Singapore) under grant 2009-08 (Edith Elkind) and by Russian Foundation for Basic Research grant 11-01-12135 ofi-m (Svetlana Obraztsova). We would like to thank the AAMAS reviewers for their very useful feedback.

9. REFERENCES

[1] J. Bartholdi and J. Orlin. Single transferable vote resists strategic voting. *Social Choice and Welfare*, 8(4):341–354, 1991.

[2] J. Bartholdi, C. Tovey, and M. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6(3):227–241, 1989.

[3] S. Brams and P. Fishburn. Voting procedures. In K. Arrow, A. Sen, and K. Suzumura, editors, *Handbook of Social Choice and Welfare, Volume 1*, pages 173–236. Elsevier, 2002.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, 2001.

[5] P. Diakonis and R. Graham. Spearman footrule as a measure of disarray. *Journal of the Royal Statistical Society B (Methodological)*, 39(2):262–268, 1977.

[6] B. Dorn and I. Schlotter. Multivariate complexity analysis of swap bribery. In *IPEC'10*, pages 107–122, 2010.

[7] E. Elkind and P. Faliszewski. Approximation algorithms for campaign management. In *WINE'10*, pages 473–482, 2010.

[8] E. Elkind, P. Faliszewski, and A. Slinko. Swap bribery. In *SAGT'09*, pages 299–310, 2009.

[9] E. Ephrati and J. Rosenschein. A heuristic technique for multi-agent planning. *Annals of Mathematics and Artificial Intelligence*, 20(1–4):13–67, 1997.

[10] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[11] A. Gibbard. Manipulation of voting schemes. *Econometrica*, 41(4):587–601, 1973.

[12] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 1938.

[13] R. Kumar and S. Vassilvitskii. Generalized distances between rankings. In *WWW'10*, pages 571–580, 2010.

[14] D. McGarvey. A theorem on the construction of voting paradoxes. *Econometrica*, 21(4):608–610, 1953.

[15] S. Obraztsova and E. Elkind. On the complexity of voting manipulation under randomized tie-breaking. In *IJCAI'11*, pages 319–324, 2011.

[16] S. Obraztsova, E. Elkind, and N. Hazon. Ties matter: Complexity of voting manipulation revisited. In *AAMAS'11*, pages 71–79, 2011.

[17] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *STOC'97*, pages 475–484, 1997.

[18] M. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10(2):187–217, 1975.

[19] I. Schlotter, P. Faliszewski, and E. Elkind. Campaign management under approval-driven voting rules. In *AAAI'11*, pages 726–731, 2011.

[20] C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904.

[21] L. Xia and V. Conitzer. Determining possible and necessary winners given partial orders. *Journal of Artificial Intelligence Research*, 41:25–67, 2011.

Manipulation Under Voting Rule Uncertainty

Edith Elkind
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
eelkind@ntu.edu.sg

Gábor Erdélyi
University of Siegen
Germany
erdelyi@wiwi.uni-siegen.de

ABSTRACT

An important research topic in the field of computational social choice is the complexity of various forms of dishonest behavior, such as manipulation, control, and bribery. While much of the work on this topic assumes that the cheating party has full information about the election, recently there have been a number of attempts to gauge the complexity of non-truthful behavior under uncertainty about the voters' preferences. In this paper, we analyze the complexity of (coalitional) manipulation for the setting where there is uncertainty about the voting rule: the manipulator(s) know that the election will be conducted using a voting rule from a given list, and need to select their votes so as to succeed no matter which voting rule will eventually be chosen. We identify a large class of voting rules such that arbitrary combinations of rules from this class are easy to manipulate; in particular, we show that this is the case for single-voter manipulation and essentially all easy-to-manipulate voting rules, and for coalitional manipulation and k-approval. While a combination of a hard-to-manipulate rule with an easy-to-manipulate one is usually hard to manipulate—we prove this in the context of coalitional manipulation for several combinations of prominent voting rules—we also provide counterexamples showing that this is not always the case.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity;
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

General Terms

Economics, Theory

Keywords

Computational Social Choice, Manipulation, Uncertainty

1. INTRODUCTION

Voting is an established framework for making collective decisions, and as such has applications in settings that range from political elections to faculty hiring decisions, selecting the winners of singing competitions, and the design of multiagent systems. In some of these settings, the number of candidates and/or voters can be large, yet the decision needs to be made quickly. Whenever this is the case, the algorithmic complexity of, on the one hand, winner determination and, on the other hand, various forms of dishonest behavior in elections, plays an important role in the selection of a voting rule: we want the former to be as low as possible, while keeping the latter as high as possible.

Traditionally, the complexity of voting rules is studied under the full information assumption: for instance, in the single-voter manipulation problem, which is perhaps one of the most fundamental problems in the complexity-theoretic analysis of voting rules, it is assumed that the manipulator knows the set of candidates, the number and the true preferences of all honest voters, and, crucially, the voting rule. However, it is widely recognized that this assumption is not always realistic, and recently a number of papers tried to analyze the complexity of cheating in elections and/or determining the likely election winners under various forms of uncertainty about the election (see Section 1.1 for an overview).

In this paper, we study the complexity of manipulation (both by a single voter and by a coalition of voters) in settings where there is uncertainty about the voting rule itself. That is, we assume that the manipulator(s) know that the voting rule belongs to a certain (finite or infinite) family of rules $\widehat{\mathcal{F}}$, and they want to select their votes so as to ensure that their preferred candidate wins, no matter which of the rules in $\widehat{\mathcal{F}}$ is chosen.

Admittedly, in political elections the voting rule to be used is typically known before the votes are cast, and the manipulator would be well advised to fully understand the voting rule before modifying her vote. However, in other applications of voting this is not always the case. For instance, it is not unusual for a university department to ask graduate students to provide a ranking of faculty candidates; however, the graduate students are not told how the hiring committee makes its decision (anecdotally, a wide variety of voting rules can be used for this purpose). Another example is provided by conference reviewing: at some point in the decision-making process, the program committee members may be asked to rank the papers whose fate has not been decided yet; the PC chair will then aggregate the rankings in a way that has not been announced to the PC members (and may, in fact, be unknown to the PC chair when she initiates the process). In some of these settings, the voters may believe that the voting rule will be chosen from a specific family of rules: for instance, the voters may know that the rule to be used is a scoring rule, or, more narrowly, a k-Approval rule (with the value of k unknown), or a Condorcet-consistent rule (see Section 2 for definitions); the situation where the voters know the voting correspondence, but not the tie-breaking rule is also captured by this description. They may then want to select their votes so that their favorite candidate wins the election *no matter which of the voting rules in this family is chosen*.

We study the complexity of this problem for several families of

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

voting rules. We limit ourselves to the setting of voting manipulation (either by a single voter or by a coalition of voters), though one can ask the same question in the context of election control or bribery (see, e.g., [13] for the definitions and a survey of recent results for these problems). We mostly focus on families that consist of a small number (usually, two) prominent voting rules, such as Plurality, k-Approval, Borda, Copeland, Maximin and STV. Our goal is not to classify all such combinations or rules: rather, we try to illustrate the general techniques that can be used for the analysis of such settings.

One would expect a combination of easy-to-manipulate rules to be easy to manipulate, and a combination of several hard-to-manipulate rules or an easy-to-manipulate one with a hard-to-manipulate one to be hard to manipulate. Our results for classic voting rules mostly confirm this intuition, with the exception of settings where we combine a hard-to-manipulate rule with one that is very indecisive. However, we show that these results are not universal: we provide an example of two hard-to-manipulate rules whose combination is easy to manipulate, as well as an example of two easy-to-manipulate rules whose combination is hard to manipulate. While the rules used in these constructions are fairly artificial, they nevertheless illustrate interesting aspects of our problem.

1.1 Related Work

Our works fits into the stream of research on winner determination and voting manipulation under uncertainty. In the context of winner determination, perhaps the most prominent problem in this category is the possible/necessary winner problem [16], where the voting rule is public information, but, for each voter, only a partial order over the candidates in known; the goal is to determine if a candidate wins the election for *some* way (the *possible winner*) or for *every* way (the *necessary winner*) of completing the voters' preferences; a probabilistic variant of this problem has also been considered [1]. Our problem is more similar in flavor to the necessary winner problem, as the manipulator has to succeed for *all* voting rules in the family.

Uncertainty about the voting rule has been recently investigated by Baumeister et al. [5], who also consider the situation where the voting rule will be chosen from a fixed set. In contrast to our work, they assume that all voters' preferences are known, and ask if there is a voting rule that makes a certain candidate a winner with respect to these preferences; thus, in their work the manipulating party is the election authority rather than one of the voters.

Our problem is, in a sense, dual to the one considered by Conitzer et al. [7]: in their model the voting rule is known, but the preferences of some of the honest voters are (partially) unknown; they ask if the manipulator can cast a vote that improves the outcome (from his perspective) for *every* realization of the honest voters' preferences; thus, just like us, they assume an adversarial environment.

There has also been some work on settings where the *effects of the manipulator's actions* are uncertain. This is the case, for instance, for the model of safe strategic voting [19], where one voter announces a manipulative vote, and one or more voters with the same true preferences may follow suit; the original manipulator does not know how many followers he will have and needs to choose the vote so as to improve the outcome for *some* number of followers, while ensuring that the outcome does not get worse for *any* number of followers. Another example is cloning [9], where the cheating party clones one or more candidates; the voters are assumed to rank the clones of a given candidate consecutively, but the exact order of the clones in voters' preferences is unknown. Our work is most similar to the variant of this problem known as 1-CLONING, where the cheating party has to succeed *no matter how the voters order the clones*.

Finally, we remark that the idea of combining two or more voting rules has been considered in early work on computational social choice [10, 14]; however, in both of these papers, voting rules are combined in a way that is very different from our work.

2. PRELIMINARIES

Given a finite set S, we denote by $\mathcal{L}(S)$ the space of all linear orders over S. An *election* is a triple $E = (C, V, \mathcal{R})$, where $C = \{c_1, \dots, c_m\}$ is the set of *candidates*, V is the set of *voters*, $|V| = n$, and $\mathcal{R} = (R_1, \dots, R_n)$ is the *preference profile*, i.e., a collection of linear orders over C. The order R_i is called the *preference order*, or *vote*, of voter i; we will also denote R_i by $\succ_i$. When $a \succ_i b$ for some $a, b \in C$, we say that voter i *prefers* a to b. A candidate a is said to be the *top-ranked* candidate of voter i, or *receive a first-place vote* from i, if $a \succ_i b$ for all $b \in C \setminus \{i\}$.

A *voting correspondence* $\mathcal{F}$ is a mapping that, given an election $E = (C, V, \mathcal{R})$ outputs a non-empty subset $S \subseteq C$; we write $S = \mathcal{F}(E)$. The elements of the set S are called the *winners* of the election E under $\mathcal{F}$. If $|\mathcal{F}(E)| = 1$ for any election E, the mapping $\mathcal{F}$ is called a *voting rule*; whenever this is the case, we abuse notation and write $\mathcal{F}(\mathcal{R}) = c$ instead of $\mathcal{F}(\mathcal{R}) = \{c\}$. We will sometimes abuse terminology and refer to voting correspondences as voting rules.

A voting correspondence $\mathcal{F}$ is said to be *neutral* if renaming the candidates does not alter the set of winners: that is, for any election $E = (C, V, \mathcal{R})$ and any permutation π of the set C, the election E' obtained by replacing each candidate c in $\mathcal{R}$ by $\pi(c)$ satisfies $\mathcal{F}(E') = \{\pi(c) \mid c \in \mathcal{F}(E)\}$. $\mathcal{F}$ is said to be *monotone* if promoting a winning candidate does not make him lose the election: if $c \in \mathcal{F}(E)$, then $c \in \mathcal{F}(E')$, where E' is obtained from E by swapping c with the candidate ranked just above c in some vote (this notion of monotonicity is sometimes referred to as *weak* monotonicity).

Voting rules We will now describe the voting rules (correspondences) considered in this paper. For all rules that assign scores to candidates (i.e., scoring rules, Copeland, and Maximin), the winners are the candidates with the highest scores.

Scoring rules Any vector $\alpha = (\alpha_1, \dots, \alpha_m) \in \mathbb{R}^m$ such that $\alpha_1 \geq \dots \geq \alpha_m$ defines a *scoring rule* $\mathcal{F}_\alpha$ over a set of candidates of size m: a candidate receives α_j points from each voter who ranks him in the j-th position, and the score of a candidate is the total number of points he receives from all voters. The vector α is called a *scoring vector*. We assume without loss of generality that the entries of α are nonnegative integers given in binary. As we require voting rules to be defined for any number of candidates, we will consider *families* of scoring rules: one for every possible number of candidates. We denote such families by $\{\mathcal{F}_{\alpha^m}\}_{m=1,\dots}$, where $\alpha^m = (\alpha_1^m, \dots, \alpha_m^m)$ is the scoring vector of length m. Two well-known examples of such families are Borda, given by $\alpha^m = (m-1, \dots, 1, 0)$ for all $m > 1$, and k-Approval, given by $\alpha_i^m = 1$ if $i \leq k$, $\alpha_i^m = 0$ if $i > k$. The 1-Approval rule is also known as Plurality.

Condorcet We say that a candidate a wins a *pairwise election* against b if more than half of the voters prefer a to b; if exactly half of the voters prefer a to b, then a is said to *tie* his pairwise election against b. A candidate is said to be a *Condorcet winner* if he wins pairwise elections against all other candidates. The Condorcet rule outputs the Condorcet winner if it exists; otherwise, it outputs the set of all candidates (recall that a voting correspondence should always output a non-empty set of winners).

Copeland Given a rational value $\alpha \in [0, 1]$, under the $\mathsf{Copeland}^\alpha$ rule each candidate gets 1 point for each pairwise election he wins and α points for each pairwise election he ties.

Maximin The Maximin score of a candidate $c \in C$ is equal to the number of votes he gets in his worst pairwise election, i.e., $\min_{d \in C \setminus \{c\}} |\{i \mid c \succ_i d\}|$.

STV Under the STV rule, the election proceeds in rounds. During each round, the candidate with the lowest Plurality score is eliminated, and the candidates' Plurality scores are recomputed. The winner is the candidate that survives till the end. If several candidates have the lowest Plurality score (we will refer to this situation as an *intermediate tie*), we assume that the candidate to be eliminated is chosen according to the lexicographic order over the candidates: if S is the set of candidates that have the lowest Plurality score in some round, we eliminate the candidate c_j such that $j \geq i$ for all $c_i \in S$. We remark that STV, as defined here, always has a single winner; however, because of the lexicographic tie-breaking rule it is not neutral.

3. PROBLEM STATEMENT

We assume that we are given a collection $\widehat{\mathcal{F}} = \{\mathcal{F}_i\}_{i \in I}$ of voting correspondences. The set $\widehat{\mathcal{F}}$ can be finite of infinite; for instance, $\widehat{\mathcal{F}}$ can be the set of all (families of) scoring rules, in which case it is infinite. When $\widehat{\mathcal{F}}$ is infinite, we assume that it admits a succinct description; if $\widehat{\mathcal{F}}$ is finite, it is assumed to be listed explicitly.

We consider the complexity of (coalitional) manipulation in elections when the manipulator does not know which of the voting rules in $\widehat{\mathcal{F}}$ will be selected. We state our definitions in the *unique winner* model, i.e., we assume that the manipulator's goal is to make its preferred candidate the unique winner with respect to each of the voting correspondences in $\widehat{\mathcal{F}}$; however, most of our results remain true in the *co-winner* model, where the manipulator would like to ensure that its preferred candidate is one of the winners under each of the voting correspondences in $\widehat{\mathcal{F}}$.

Name: $\widehat{\mathcal{F}}$-MANIPULATION BY SINGLE VOTER (SM)

Input: An election (C, V) with $|C| = m$, $|V| = n - 1$, a preference profile $\mathcal{R} = (R_1, \ldots, R_{n-1})$, and a candidate $p \in C$.

Question: Is there a vote $L \in \mathcal{L}(C)$ such that p is the unique winner in $(\mathcal{R}, L)$ with respect to each of the voting correspondences in $\widehat{\mathcal{F}}$?

Voters $1, \ldots, n - 1$ are referred to as the *honest* voters, and the last voter (the one who submits vote L and wants p to win) is referred to as the *manipulator*.

Name: $\widehat{\mathcal{F}}$-COALITIONAL MANIPULATION (CM)

Input: An election (C, V) with $|C| = m$, $|V| = h$, a set M, $|M| = s = n - h$, a preference profile $\mathcal{R} = (R_1, \ldots, R_h)$, and a candidate $p \in C$.

Question: Is there a profile $\mathcal{L} = (L_1, \ldots, L_s) \in \mathcal{L}^s(C)$ such that p is the unique winner in $(\mathcal{R}, \mathcal{L})$ with respect to each of the voting correspondences in $\widehat{\mathcal{F}}$?

If $\widehat{\mathcal{F}}$ is finite, we say that an algorithm $\mathcal{A}$ for $\widehat{\mathcal{F}}$-SM or $\widehat{\mathcal{F}}$-CM is a polynomial-time algorithm if its running time is polynomial in n, m, and $|\widehat{\mathcal{F}}|$; if $\widehat{\mathcal{F}}$ is infinite, we require the running time of $\mathcal{A}$ to be polynomial in n and m. We remark that $\widehat{\mathcal{F}}$-SM (respectively, $\widehat{\mathcal{F}}$-CM) is in NP for any finite collection $\widehat{\mathcal{F}}$ of polynomially computable voting rules: it suffices to guess a manipulative vote L (respectively, a list $(L_1, \ldots, L_s)$ of manipulative votes) and verify that it makes p the unique winner under every rule in $\widehat{\mathcal{F}}$. Thus, in what follows, when proving that these problems are NP-complete for some finite $\widehat{\mathcal{F}}$, we will only provide an NP-hardness proof.

Traditionally, the problems $\widehat{\mathcal{F}}$-SM and $\widehat{\mathcal{F}}$-CM are studied for the case $|\widehat{\mathcal{F}}| = 1$. In what follows, whenever $\widehat{\mathcal{F}} = \{\mathcal{F}\}$, we omit the curly braces and write $\mathcal{F}$-SM/CM instead of $\{\mathcal{F}\}$-SM/CM to conform with the standard notation.

4. MANIPULATION

We start by considering the SM problem. In their classic paper [3], Bartholdi, Tovey and Trick show that this problem is polynomial-time solvable for $\mathsf{Copeland}^\alpha$ (for every rational $\alpha \in [0, 1]$), Maximin, and all scoring rules (while Bartholdi et al. do not explicitly consider scoring rules other than Plurality and Borda, it is not hard to see that their algorithm works for any scoring rule).

Remarkably, for all these rules the manipulative vote can be found by essentially the same algorithm. This algorithm starts by ranking p first; it is safe to do so, because all of these rules are monotone. Note that at this point we can already compute p's final score; let us denote it by $s(p)$. The algorithm then fills up positions $2, \ldots, m$ in the vote one by one. When considering position i, $i \geq 2$, it tries to place each of the still unranked candidates into this position. At this point, the identities of the candidates in positions $1, \ldots, i - 1$ are already known, so one can determine the score of each candidate c if it were to be placed in position i (this is true for Copeland, Maximin and all scoring rules, but need not be true in general, even for monotone rules); let us denote this quantity by $s_i(c)$. If there exists a candidate c such that $s_i(c) < s(p)$, it is placed in position i; if there are several such candidates, one of them is selected arbitrarily. If no such candidate can be found, the algorithm reports that no manipulative vote exists.

Bartholdi et al. prove the correctness of this algorithm for all voting correspondences that (1) are monotone and (2) have the property that the score of a candidate c can be determined if we know which candidates are ranked above and below c in each vote, and the winners are the candidates with the highest score. $\mathsf{Copeland}^\alpha$, $\alpha \in \mathbb{Q} \cap [0, 1]$, Maximin, and all scoring rules satisfy both of these conditions, and STV satisfies neither of them; indeed, STV-SM is known to be NP-complete [2].

We will now show that the algorithm of Bartholdi et al. extends to $\widehat{\mathcal{F}}$-SM for any finite set $\widehat{\mathcal{F}}$ that consists of voting correspondences that satisfy (1) and (2).

THEOREM 4.1. *Let $\widehat{\mathcal{F}}$ be a finite set of voting rules such that every rule $\mathcal{F}_i \in \widehat{\mathcal{F}}$ satisfies conditions (1) and (2). Then $\widehat{\mathcal{F}}$-SM can be solved in polynomial time.*

PROOF. Let $\widehat{\mathcal{F}} = \{\mathcal{F}_1, \ldots, \mathcal{F}_\ell\}$. Our algorithm proceeds in rounds: in round i, $i = 1, \ldots, m$, we consider position i.

In the first round, we place p in the top position; let $s^j(p)$ denote p's score with respect to the rule $\mathcal{F}_j$, $j = 1, \ldots, \ell$, in the resulting election.

Now, consider round i, $i = 2, \ldots, m$. For each candidate c that has not been ranked in round $1, \ldots, i - 1$, let $s^j_i(c)$ be his score under rule $\mathcal{F}^j$ if he were to be ranked in position i at this point. If there exists an unranked candidate c such that $s^j_i(c) < s^j(p)$ for all $j = 1, \ldots, \ell$, we place c in position i; if there are several such candidates, we choose one of them arbitrarily. If no such candidate exists, we report that the input instance of $\widehat{\mathcal{F}}$-SM cannot be manipulated. If we manage to successfully rank all candidates, we report that there exists a successful manipulative vote; in fact, our algorithm constructs it.

Clearly, if our algorithm reports that a manipulative vote exists, this is indeed the case. Conversely, suppose that our algorithm reports that the given election cannot be manipulated. This means that during some round i, we have $s_i^j(c) \geq s^j(p)$ for some voting rule $\mathcal{F}_j \in \widehat{\mathcal{F}}$ and every candidate c that has not been ranked in rounds $1, \dots, i-1$. But then consider the execution of the original algorithm of Bartholdi et al. [3] on the instance of $\mathcal{F}_j$-SM given by the same election. The algorithm of Bartholdi et al. could have made exactly the same choices as our algorithm in rounds $1, \dots, j-1$. Therefore, it, too, would have reported that its input instance is a "no"-instance. Since the algorithm of Bartholdi et al. is known to be correct, this means that no manipulative vote could have made p the unique winner with respect to $\mathcal{F}_j$. Hence, our instance of $\widehat{\mathcal{F}}$-SM is a "no"-instance as well, which means that our algorithm is correct. □

The proof of Theorem 4.1 is very simple. However, the result itself plays a key role in our understanding of single-voter manipulation under voting rule uncertainty. Indeed, to the best of our knowledge, for all classic voting rules for which single-voter manipulation is known to be easy, a manipulative vote can be constructed using the algorithm of [3]. Therefore, we cannot hope to put together two or more classic easy-to-manipulate rules so that the manipulation problem with respect to the combination of these rules is computationally hard.

One can nevertheless ask if such a combination of rules exists. We will now show that the answer to this question is "yes": we present two easy-to-manipulate rules, which we will call STV_1 and STV_2, such that STV_i-SM is polynomial-time solvable for $i = 1, 2$ but $\{\mathsf{STV}_1, \mathsf{STV}_2\}$-SM is NP-hard. Admittedly, these rules are not particularly natural; but then Theorem 4.1 shows that we cannot hope to prove a result of this type for natural voting rules.

The main idea of the construction is that each of these rules can be manipulated either by making p the STV winner or by using an easy-to-compute "trapdoor"; however, the "trapdoors" for STV_1 and STV_2 are incompatible with each other, so, to manipulate both, one needs to manipulate STV.

Formally, STV_1 is defined as follows. For $m \leq 3$, all candidates are declared to be the winners. For $m > 3$, the rule is not neutral in a very essential way: candidates c_{m-2}, c_{m-1} and c_m play a special role. Specifically, if some voter ranks c_{m-3+j} in position $m-3+j$ for $j = 1, 2, 3$, then the candidate ranked first by this voter is declared to be the election winner; if there are several such voters, the set of winners consists of these voters' top choices. Otherwise, the winner is the winner under the STV rule.

STV_2 coincides with STV_1 for $m \leq 3$. For $m > 3$, if some voter ranks c_{m-3+j} in position c_{m+1-j} for $j = 1, 2, 3$, then the candidate ranked first by this voter is declared to be the election winner (again, the election may have multiple winners if there are several such voters), and otherwise the winner is the STV winner.

THEOREM 4.2. STV_1-SM *and* STV_2-SM *are in* P. *However,* $\{\mathsf{STV}_1, \mathsf{STV}_2\}$-SM *is* NP-*complete.*

PROOF. Consider an instance of STV_1. Suppose that some of the honest voters rank c_{m-3+j} in position $m-3+j$ for $j = 1, 2, 3$, and let S be the set of these voters' top choices. If $S \neq \{p\}$, no matter what the manipulator does, all candidates in S will be declared the election winners, so the manipulator cannot make p the unique winner. If $S = \{p\}$, or if none of the honest voters ranks c_{m-3+j} in position $m-3+j$ for $j = 1, 2, 3$, the manipulator can rank p first and place c_{m-3+j} in position $m-3+j$ for $j = 1, 2, 3$; this would make p the unique winner. In any case, the manipulator's problem is in P. A similar argument shown that STV_2-SM is in P.

To show that $\{\mathsf{STV}_1, \mathsf{STV}_2\}$-SM is NP-hard, we will provide an NP-hardness reduction from STV-SM, which is known to be NP-complete [2].

Given an instance of STV-SM with a set of candidates $C = \{c_1, \dots, c_{m'}\}$, a set of voters V, $|V| = n-1$, a preference profile $\mathcal{R} = (R_1, \dots, R_{n-1})$ over C, and a preferred candidate $p \in C$, we will modify it as follows. We let $m = m' + 3$ and set $C' = C \cup \{c_{m-2}, c_{m-1}, c_m\}$. We ask each of the voters to rank each of the candidates in C in the same position as before, and rank c_{m-1} in position $m-2$, followed by c_{m-2} and c_m; denote the resulting preference profile by $\mathcal{R}'$.

Observe that the manipulator can make p the unique winner of this election under STV_1 either by ranking c_{m-3+j} in position $m-3+j$ for $j = 1, 2, 3$, or by making p the unique STV winner. Similarly, the manipulator can make p the unique winner of the new election under STV_2 either by ranking c_{m-3+j} in position $m+1-j$ for $j = 1, 2, 3$, or by making p the unique STV winner.

Now, suppose that the original instance of STV-SM is a "yes"-instance, and let $L \in \mathcal{L}(C)$ be the manipulative vote that makes p the STV winner in that election. Consider the vote L' obtained from L by ranking c_{m-1}, c_{m-2}, and c_m after all candidates in C (in this order). In $(\mathcal{R}', L')$, no voter ranks c_{m-2}, c_{m-1}, c_m according to either of the "trapdoors", so both in STV_1 and in STV_2 the STV rule is applied. Further, in $(\mathcal{R}', L')$ candidates c_{m-2}, c_{m-1}, c_m receive no first-place votes, so under STV they are eliminated before any candidates in C. STV then proceeds in the same way as on $(\mathcal{R}, L)$, thus making p the winner.

Conversely, suppose that there exists a vote $L' \in \mathcal{L}(C')$ such that p is the unique winner in $(\mathcal{R}', L')$ with respect to both STV_1 and STV_2. Since L cannot rank c_m in positions $m-2$ and m simultaneously, it follows that p is the STV winner in $(\mathcal{R}', L')$. Now, consider the execution of STV_1 on $(\mathcal{R}', L')$. If L' does not rank any of the candidates in $C' \setminus C$ in the top position, after the first three steps the execution of STV_1 on $(\mathcal{R}', L')$ coincides with the execution of STV on $(\mathcal{R}, L)$, where L is obtained from L' by removing c_{m-2}, c_{m-1} and c_m. Thus, in this case L is a successful manipulative vote that witnesses that the original instance of STV-SM is a "yes"-instance.

Now, suppose that L' ranks a candidate from $C' \setminus C$ first; assume without loss of generality that the top candidate in L is c_m. Then simply removing c_{m-2}, c_{m-1} and c_m from L' would not necessarily work: if the top candidate in the resulting vote receives no first-place votes in $\mathcal{R}$, this candidate would have been eliminated in the very beginning in $(\mathcal{R}', L')$, but may survive much longer in the modified election. Thus, we need a slightly different strategy. Let C_0 be the set of candidates that receive no first-place votes in $\mathcal{R}$. We construct L from L' by removing c_{m-2}, c_{m-1} and c_m and moving candidates in C_0 to the bottom of the vote (without changing the relative ordering of all other candidates). Then on $(\mathcal{R}', L')$ STV starts by eliminating c_{m-1}, c_{m-2} and the candidates in C_0. At this point, each candidate has at least one first-place vote; hence, because of our intermediate tie-breaking rule, c_m is the first candidate to be eliminated, and we are left with an election E'' over $C \setminus C_0$. On the other hand, in $(\mathcal{R}, L)$ the set of candidates with no first-place votes coincides with C_0, so after the first $|C_0|$ elimination rounds we obtain an election over $C \setminus C_0$ that coincides with E''. Hence, p is the unique STV winner in $(\mathcal{R}, L)$, and hence our original instance of STV-SM is a "yes"-instance. □

We remark that Theorem 4.2 holds for coalitional manipulation as well: for the easiness result, note that the manipulators may use trapdoors to manipulate STV_1 or STV_2, and the hardness result generalizes trivially.

The next question that we would like to explore is whether a combination of an easy-to-manipulate rule with a hard-to-manipulate one is hard to manipulate. We will now illustrate that this is the case for two classic voting rules, namely, STV and Borda.

THEOREM 4.3. $\{\text{Borda}, \text{STV}\}$-SM *is* NP-*complete.*

PROOF. We will provide a reduction from STV-SM. Consider an instance of STV-SM given by an election (C, V) with $C = \{c_1, \dots, c_m\}$, $|V| = n-1$, a preference profile $\mathcal{R} = (R_1, \dots, R_{n-1})$, and a candidate $p \in C$; assume without loss of generality that $n \geq 3$. Suppose that p is not ranked first by any of the voters in V. Then if the manipulator does not rank p first, p get eliminated before any candidate that has a positive Plurality score in (C, V) and therefore does not win the election. Hence, the manipulator has to rank p first. Observe also that the rest of the manipulator's vote does not matter in this case: it can only impact the candidate elimination process after p is eliminated, at which point p has already lost the election. Thus, if no voter in V ranks p first, the manipulator's problem is in P: the manipulator should rank p first and check if this achieves the desired result. We can therefore assume without loss of generality that in our input instance of STV-SM candidate p receives at least one first-place vote.

Thus, assume that p is the top candidate of voter 1. Let $D = \{c_{im+j} \mid i = 1, \dots, n, j = 1, \dots, m\}$, and set $C' = C \cup D$. Modify all votes in $\mathcal{R}$ by inserting the candidates in D right below p in each vote, in an arbitrary order; let $\mathcal{R}'$ be the resulting profile.

Let $s(c)$ denote the Borda score of a candidate $c \in C$ in $(C, V, \mathcal{R})$, and let $s'(c)$ denote his score in $(C', V, \mathcal{R}')$. We have $s(c) \leq (n-1)(m-1)$ for all $c \in C$. Moreover, we have $s'(p) = s(p) + mn(n-1)$, as p gets mn extra points from each vote. On the other hand, every other candidate in C gets at most $mn(n-2)$ extra points from voters $2, \dots, n-1$ and no extra points from voter 1. Thus, for any $c \in C \setminus \{p\}$ we have

$$s'(c) \leq s(c) + mn(n-2) \leq mn(n-1) - m - n + 1 < s'(p) - m.$$

Also, the Borda score of any $d \in D$ in $(C', V, \mathcal{R}')$ is less than $s'(p)$. Thus, if the manipulator ranks the candidates in C in top m positions, p is the unique Borda winner of the resulting election.

On the other hand, no matter how the manipulator votes, under STV all candidates in D will be eliminated before all candidates in C that have a non-zero Plurality score: indeed, the Plurality score of each $d \in D$ is at most 1, and the intermediate tie-breaking rule favors candidates in C over those in D.

We are now ready to show that our reduction is correct. Let L be a successful manipulative vote for the original instance, and let C_0 be the set of all candidates in C with no first-place votes in $(\mathcal{R}, L)$. Note that the candidates in C_0 are eliminated in the first $|C_0|$ rounds of STV. Now, consider the vote L' obtained from L by ranking the candidates in D in positions $m+1, \dots, m(n+1)$. In the election $(\mathcal{R}', L')$ candidate p has the highest Borda score. Moreover, under STV we will first eliminate all candidates in $C_0 \cup D$. At this point, we obtain the same election as after $|C_0|$ rounds of STV on $(\mathcal{R}, L)$—and hence the same winner. Thus, L' is a successful manipulative vote in the new election.

Conversely, suppose that $L' \in \mathcal{L}(C \cup D)$ is such that in $(\mathcal{R}', L')$ candidate p is both the unique Borda winner and the (unique) STV winner. Let C'_0 be the set of candidates in C that have no first-place votes in $(\mathcal{R}', L')$. When we execute STV on $(\mathcal{R}', L')$, we eliminate all candidates in $D \cup C'_0$ prior to eliminating any of the candidates in $C \setminus C'_0$. Let L be the vote in $\mathcal{L}(C)$ obtained by deleting all candidates in D from L' and moving all candidates in C'_0 to the bottom $|C'_0|$ positions (without changing the relative ordering of the candidates in $C \setminus C'_0$). Then C'_0 is exactly the set of candidates in C who have no first-place votes in $(\mathcal{R}, L)$. Therefore, when we execute STV on $(\mathcal{R}, L)$, we eliminate all candidates in C'_0 prior to eliminating any candidates in $C \setminus C'_0$. Thus, the profile obtained after running STV for $|D| + |C'_0|$ steps on $(\mathcal{R}', L')$ coincides with the profile obtained after running STV for $|C'_0|$ steps on $(\mathcal{R}, L)$. Thus, L is a successful manipulative vote for the original election. $\square$

Another interesting (and arguably natural) combination of voting rules is $\{\text{Plurality}, \text{STV}\}$. Here, we were unable to provide a black-box reduction showing that the combination of these rules is hard to manipulate. However, a careful inspection of Bartholdi and Orlin's proof [2] establishes that $\{\text{Plurality}, \text{STV}\}$-SM is indeed NP-hard: by tweaking the instance of STV constructed in that proof we can ensure that the manipulator's preferred candidate is the unique Plurality winner.

However, there are also examples where the combination of a hard-to-manipulate rule and an easy-to-manipulate one is easy to manipulate. Consider, for instance, the following rule: if some candidate receives strictly more than $\lfloor n/2 \rfloor$ first-place votes, he is the unique election winner; otherwise, all candidates are winners. We will refer to this rule as the Majority rule. Majority is not particularly decisive, but apart from that it is a reasonable voting rule. Clearly, it is easy to manipulate: the manipulator simply needs to check if ranking p first does the job. Moreover, the combination of Majority and STV is easy to manipulate, too.

THEOREM 4.4. $\{\text{Majority}, \text{STV}\}$-SM *is in* P.

PROOF. Consider an election $E = (C, V, \mathcal{R})$. If in this election p is ranked first by at most $\lfloor n/2 \rfloor - 1$ voters, the manipulator cannot make p the Majority winner, so this is a "no"-instance of our problem. On the other hand, if p is ranked first by at least $\lfloor n/2 \rfloor$ voters, the manipulator can rank p first, making him both the unique Majority winner and the unique STV winner. $\square$

The reason why the combination of Majority and STV is easy to manipulate is that Majority is always guaranteed to elect the STV winner: if some candidate has more than $\lfloor n/2 \rfloor$ votes, he will obviously win under STV, and in all other cases Majority elects all candidates. Using this observation, we can now generalize Theorem 4.4. We will say that a voting correspondence $\mathcal{F}_1$ is a *refinement* of a voting correspondence $\mathcal{F}_2$ if for any election E we have $\mathcal{F}_1(E) \subseteq \mathcal{F}_2(E)$, and there exists an election for which this containment is strict. Now, it is easy to see that STV is a refinement of Majority. Also, some of the voting rules defined in Section 2 are refinements of each other: namely, both Copeland and Maximin are refinements of Condorcet. Yet another example is provided by the so-called second-order Copeland rule, proved to be NP-hard to manipulate in [3]: this rule is obtained by combining the Copeland rule with a rather sophisticated tie-breaking rule, and is therefore a refinement of Copeland. Now, it is easy to see that the proof of Theorem 4.4 implies a more general fact.

COROLLARY 4.5. *If a voting correspondence $\mathcal{F}_1$ is a refinement of a voting correspondence $\mathcal{F}_2$ and $\mathcal{F}_2$-SM is in* P*, then so is* $\{\mathcal{F}_1, \mathcal{F}_2\}$-SM.

We remark that Corollary 4.5 crucially relies on the fact that we consider the unique-winner version of SM, and the requirement that a voting correspondence should produce a non-empty set of winners for every election. Also, the converse of Corollary 4.5 is not true, as illustrated by Copeland and second-order Copeland. Another important observation is that Corollary 4.5 applies equally well to the coalitional manipulation problem; we will make use of this fact in Section 5.

Now, suppose we have two hard-to manipulate rules. Clearly, it can be the case that their combination is also hard to manipulate: for example we can take two copies of STV (if we insist that these two rules should be distinct, we can modify one of the copies to produce a different winner on a single profile; this does not affect the complexity of our problem). To conclude this section, we provide an example of two voting rules $\mathcal{F}_1$ and $\mathcal{F}_2$ such that both $\mathcal{F}_1$-SM and $\mathcal{F}_2$-SM are NP-complete, but $\{\mathcal{F}_1, \mathcal{F}_2\}$-SM is in P; thus, counterintuitively, even a combination of hard-to-manipulate rules can be "easy" to manipulate (it will become clear in a minute why we used quotes in the previous sentence).

Our first voting rule is STV. Our second rule, which we will denote by STV′, is obtained from STV by the following modification: if c_i is the STV winner in E, then we output c_{i+1} as the unique winner (where $c_{m+1} := c_1$). Now, clearly, manipulating STV′ is just as hard as manipulating STV: we simply have to solve the STV manipulation problem for a different candidate. However, for any election E, STV and STV′ have different winners, so there is no way the manipulator can make p win under both of them. Thus, the manipulator's problem is "easy", in the sense that it simply cannot achieve its goal, so every instance of $\{\mathsf{STV}, \mathsf{STV}'\}$-SM is a "no"-instance. We summarize these observations as follows.

THEOREM 4.6. *STV′-SM is NP-complete. On the other hand,* $\{\mathsf{STV}, \mathsf{STV}'\}$-SM *is in* P.

We remark that Theorem 4.6 extends trivially to coalitional manipulation.

5. COALITIONAL MANIPULATION

The coalitional manipulation problem is known to be NP-hard for many prominent voting rules, such as Borda [6, 8] and some other scoring rules [20], Copeland$^\alpha$ for $\alpha \in (\mathbb{Q} \cap [0, 1]) \setminus \{0.5\}$ [11, 12] and Maximin [21]; it goes without saying that the hardness result for STV-SM [2] implies that STV-CM is NP-hard as well. Therefore, we cannot hope for a general easiness result along the lines of Theorem 4.1. Nevertheless, we can identify some interesting combinations of voting rules for which CM is in P.

We start by observing that Condorcet-CM is in P. Indeed, the manipulators can simply rank p first in all of their votes and check if that makes p the Condorcet winner; note that the answer to this question does not depend on how the manipulators rank the other candidates. Now, by extending Corollary 4.5 to the coalitional manipulation problem, and using the fact both Maximin and Copeland are refinements of the Condorcet rule, we obtain the following corollaries.

COROLLARY 5.1. $\{\mathsf{Condorcet}, \mathsf{Maximin}\}$-CM *is in* P.

COROLLARY 5.2. $\{\mathsf{Condorcet}, \mathsf{Copeland}^\alpha\}$-CM *is in* P *for any* $\alpha \in \mathbb{Q} \cap [0, 1]$.

Of course, the coalitional manipulation problem is also easy for the Majority rule, and it can be easily checked that each of the rules defined in Section 2 is a refinement of the Majority rule. Thus, we could obtain a similar easiness result for the combination of Majority and any other rule. We chose to state Corollaries 5.1 and 5.2 for the Condorcet rule, as the latter is more decisive and has been considered in prior work on computational social choice, albeit in the context of control [4].

We will now move on to another family of voting rules whose combinations can be shown to be easy to manipulate. A recent paper by Lin [18] shows that the coalitional manipulation problem is easy for k-Approval for any value of k. We will now prove a stronger statement: coalitional manipulation is easy even for combinations of k-Approval rules (for different values of k).

THEOREM 5.3. *For any finite set* $K = \{k_1, \dots, k_\ell\} \subseteq \mathbb{N}$, *the problem* $\{k_1\text{-}\mathsf{Approval}, \dots, k_\ell\text{-}\mathsf{Approval}\}$-CM *is in* P.

PROOF. Consider an election E with $C = \{c_1, \dots, c_m\}$, $|V| = h$, $|M| = s$, and $\mathcal{R} = (R_1, \dots, R_h)$. We can assume without loss of generality that $p = c_m$.

Since k-Approval is monotone for any value of k, it is optimal for the manipulators to rank p first in all s votes. For each $k \in K$, let $s_k(p)$ be p's k-Approval score in the resulting election. Now, the manipulators' goal is to rank every other candidate $c \in C \setminus \{p\}$ so that for each $k \in K$ the k-Approval score of c is strictly less than $s_k(p)$. We can assume without loss of generality that for each $k \in K$ and each $c \in C \setminus \{p\}$ the k-Approval score of c in $\mathcal{R}$ is strictly less than $s_k(p)$: otherwise, we clearly have a "no"-instance of our problem. Now, for each $r = 2, \dots, m$ and each c_j, $j = 1, \dots m-1$, let $x(r, j)$ be the maximum number of times that c_j can be ranked in position r or higher in the manipulators' votes so that its k-Approval score is less than $s_k(p)$ for every $k \in K$. These values are easy to compute from the candidates' k-Approval scores in $\mathcal{R}$, $k \in K$; our assumption on the initial scores ensures that they are non-negative.

We will now construct a flow network so that the maximum flow in this network corresponds to a successful set of manipulative votes, if one exists. Our network has a source S, a sink T, a node c_j for each $j = 1, \dots, m-1$, and a node p_r for $r = 2, \dots, m$; intuitively, node p_r corresponds to position r in the manipulators' votes. There is an edge of capacity s from S to each c_j, $j = 1, \dots, m-1$, and an edge of capacity s from each p_r, $r = 2, \dots, m$, to T. Essentially, the edge from S to c_j ensures that c_j is ranked by each manipulator, and the edge from p_r to T ensures that each of the manipulators fills position r in his vote. It remains to explain how to connect the candidates with the positions.

For each $c_j \in C \setminus \{p\}$ we build a caterpillar graph that connects c_j to $p_m, \dots, p_2$. More formally, for each candidate c_j, $j = 1, \dots, m-1$, we introduce nodes $z_{j,m}, \dots, z_{j,2}$ and edges $(c_j, z_{j,m})$, $(z_{j,r}, z_{j,r-1})$ for $r = m, \dots, 3$, and $(z_{j,r}, p_r)$ for $r = m, \dots, 2$. The capacity of $(c_j, z_{j,m})$ and $(z_{j,r}, p_r)$, $r = m, \dots, 2$, is $+\infty$, and the capacity of $(z_{j,r}, z_{j,r-1})$, $r = m, \dots, 3$, is given by $x(r-1, j)$. This completes the description of our network (see Figure 1).

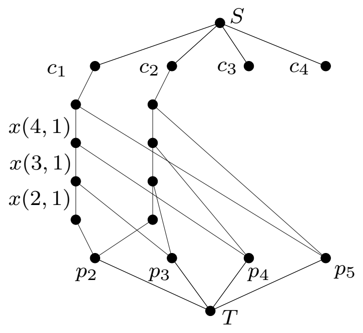


Figure 1: Network in the proof of Theorem 5.3, $m = 5$

We claim that this network admits a flow of size $s(m-1)$ if and only if there exists an assignment of candidates to the positions in the manipulators' votes such that the k-Approval score of each $c \in C \setminus \{p\}$ is less than $s_k(p)$ for every $k \in K$. Indeed, suppose that such a flow exists. Since all capacities are integer, we can assume that this flow is integer. It saturates all edges leaving S, so there are s units of flow leaving each c_j, $j = 2, \dots, m$. This flow has to reach $p_2, \dots, p_m$ traveling through the caterpillar graph associated with c_j. Thus, we can associate the flow on the edge $(z_{j,r}, p_r)$ with the number of times that c_j is ranked in position p_r. The capacity constraints on edges guarantee that these numbers correspond to a valid set of manipulators' votes. Moreover, for each $r = m, \dots, 2$, the total flow from c_j to $p_r, \dots, p_2$ is at most $x(r, j)$, which ensures that c_j is ranked in positions $p_r, \dots, p_2$ at most $x(r, j)$ times. Hence, for each $k \in K$ and each $j = 1, \dots, m-1$, the k-Approval score of c_j is less than that of p, and therefore p is the unique winner under each of the rules in our collection. Conversely, a vote that makes p the unique election winner with respect to each k-Approval, $k \in K$, can be converted into a valid flow; if x manipulators rank c_j in position r, we send x units of flow on $(z_{j,r}, p_r)$. $\square$

Theorem 5.3 has an interesting implication. Let $\widehat{\mathcal{F}}_\alpha$ be the family of all scoring rules. Observe that $\widehat{\mathcal{F}}_\alpha$ includes the Borda rule, for which coalitional manipulation is hard. Nevertheless, it turns out that $\widehat{\mathcal{F}}_\alpha$-CM is solvable in polynomial time.

THEOREM 5.4. *$\widehat{\mathcal{F}}_\alpha$-CM is in* P.

PROOF. We will use the following folklore observation [17]: a candidate c is the unique winner of an election $E = (C, V, \mathcal{R})$ with respect to each $|C|$-candidate scoring rule if and only if c is the unique k-Approval winner of E for $k = 1, \dots, |C| - 1$. Thus, to solve $\widehat{\mathcal{F}}_\alpha$-CM on an instance with m candidates, it suffices to apply the algorithm described in the proof of Theorem 5.3 with $K = \{1, \dots, m-1\}$. Clearly, the running time of this algorithm is polynomial in n and m. $\square$

We will now provide several examples of combinations of rules for which coalitional manipulation is hard. We will focus on classic voting rules, and investigate combinations of the most prominent easy-to-manipulate rule, namely, Plurality, with Borda and Copeland, which are both hard for coalitional manipulation.

THEOREM 5.5. *{Plurality, Borda}-CM is* NP*-complete.*

PROOF. Similarly to the proofs in Section 4, we will start with an instance of Borda-CM; this problem is known to be NP-hard even for two manipulators and three input votes [8]. Consider an instance of Borda-CM with $C = \{c_1, \dots, c_m\}$, $|V| = 3$, $\mathcal{R} = (R_1, R_2, R_3)$ and $|M| = 2$; assume without loss of generality that $m \geq 8$. Since both Borda and Plurality are neutral, we can assume without loss of generality that $p = c_m$.

For each $i = 1, \dots, m-1$, let X_i be an arbitrary vote in $\mathcal{L}(C)$ that ranks p first and c_i last, and let X'_i be the vote obtained by reversing X_i (i.e., if in X_i candidate c is ranked above candidate d, then in X'_i candidate d is ranked above c, for any $c, d \in C$). We modify the input election by adding votes X_i and X'_i for all $i = 1, \dots, m-1$; denote the resulting election by $\mathcal{R}'$. Note that this increases p's Plurality score by $m-1$, but the Plurality score of any other candidate only increases by 1. On the other hand, the Borda score of each candidate increases by exactly $(m-1)^2$.

Suppose we have started with a "yes"-instance of Borda-CM, and let L_1, L_2 be the manipulators' votes such that p is the unique Borda winner of $(\mathcal{R}, L_1, L_2)$. Clearly, p is also the unique Borda winner of $(\mathcal{R}', L_1, L_2)$. Moreover, the Plurality score of p in $(\mathcal{R}', L_1, L_2)$ is at least $m - 1 \geq 7$, while the Plurality score of any other candidate is at most $|V| + |M| + 1 = 6$, so p is also the unique Plurality winner in $(\mathcal{R}', L_1, L_2)$ (the careful reader will notice that we can relax the requirement that $m \geq 8$ by observing that Borda is monotone). Conversely, suppose that our instance of {Plurality, Borda}-CM is a "yes"-instance. Then there exist some votes $L'_1, L'_2 \in \mathcal{L}(C)$ that make p the unique Borda winner of $(\mathcal{R}', L'_1, L'_2)$. But then p is also the unique Borda winner of $(\mathcal{R}, L_1, L_2)$. $\square$

It is interesting to compare Theorem 5.4 and Theorem 5.5: the former implies that the combination of Borda with *all* k-Approval rules is easy to manipulate, whereas the latter shows that the combination of 1-Approval (i.e., Plurality) and Borda is hard to manipulate; we remark that the proof of Theorem 5.5 extends easily to the combination of Borda with k-Approval for any constant k.

A construction similar to the one used in the proof of Theorem 5.5 shows that {Plurality, Copeland$^\alpha$}-CM is NP-complete for $\alpha \in (\mathbb{Q} \cap [0,1]) \setminus \{0.5\}$ (this is the range of values of α for which Copeland$^\alpha$-CM in known to be NP-complete). The only difference is that for Copeland we cannot assume that the number of voters is a small constant (we will, however, assume that there are exactly two manipulators, as this is known to be sufficient for the NP-hardness of this problem [11, 12]). Therefore, instead of adding one pair (X_i, X'_i) for each $i = 1, \dots, m-1$, we add h such pairs, where h is the number of honest voters. This modification has no impact on Copeland scores: if c beats d in the original profile, this remains to be the case when the new votes are added; the converse is also true. However, the Plurality score of p increases by $h(m-1)$, whereas the Plurality score of any other candidate increases by h, and, as a result, does not exceed $2h + 2$ (even taking the manipulators' votes into account). Assuming without loss of generality that $m \geq 4$ and $h \geq 3$, we obtain that p is the unique Plurality winner of the modified election, irrespective of how the manipulator votes. The rest of the argument proceeds as in the proof of Theorem 5.5. We obtain the following corollary.

COROLLARY 5.6. *{Plurality, Copeland$^\alpha$}-CM is* NP*-complete for $\alpha \in (\mathbb{Q} \cap [0,1]) \setminus \{0.5\}$.*

Perhaps unsurprisingly, the combination of Borda and Copeland is hard to manipulate as well.

THEOREM 5.7. *{Borda, Copeland$^\alpha$}-CM is* NP*-complete for $\alpha \in (\mathbb{Q} \cap [0,1]) \setminus \{0.5\}$.*

PROOF. We employ a variant of the construction used in the proof of Corollary 5.6: we start with an instance of Copeland$^\alpha$-CM with $C = \{c_1, \dots, c_m\}$, $|V| = h$, $|M| = 2$, and $\mathcal{R} = (R_1, \dots, R_h)$ and modify it to obtain an instance of our problem in which p is the Borda winner no matter how the manipulator votes.

We assume without loss of generality that $h > 2$. Let $C' = C \cup \{d\}$, and modify $\mathcal{R}$ by ranking d in the last position in each preference order; denote the resulting profile by $\mathcal{R}'$. In $\mathcal{R}'$, d loses all pairwise elections, no matter how the manipulator votes; moreover, the final Copeland$^\alpha$ scores of all candidates do not depend on how the manipulators rank d.

Now, let X be some vote in $\mathcal{L}(C \cup \{d\})$ that ranks p first and d second, let X' be obtained by reversing X, and let X'' be obtained from X' by swapping p and d. Add $2mh$ pairs of the form (X, X'') to $\mathcal{R}'$; denote the resulting profile by $\mathcal{R}''$. Clearly, the addition of these new votes cannot possibly change the outcome of any pairwise election other than the one between p and d; moreover, p won his pairwise election against d even before these new votes were added, so this remains to be the case. We conclude that the Copeland$^\alpha$ score of any candidate $c \in C$ in $\mathcal{R}''$ exceeds his Copeland$^\alpha$ score in $\mathcal{R}$ by exactly 1.

	SM	CM
easy + easy = easy	all “nice” rules	k-Approval
easy + easy = hard	$\{\mathsf{STV}_1, \mathsf{STV}_2\}$	$\{\mathsf{STV}_1, \mathsf{STV}_2\}$
easy + hard = hard	$\{\mathsf{Borda}, \mathsf{STV}\}$, $\{\mathsf{Plurality}, \mathsf{STV}\}$	$\{\mathsf{Plurality}, \mathsf{Borda}\}$, $\{\mathsf{Plurality}, \mathsf{Copeland}\}$
easy + hard = easy	$\{\mathsf{Majority}, \mathsf{STV}\}$	$\{\mathsf{Condorcet}, \mathsf{Copeland}\}$, $\{\mathsf{Condorcet}, \mathsf{Maximin}\}$, scoring rules
hard + hard = easy	$\{\mathsf{STV}, \mathsf{STV}'\}$	$\{\mathsf{STV}, \mathsf{STV}'\}$
hard + hard = hard	$\{\mathsf{STV}, \mathsf{STV}\}$	$\{\mathsf{Borda}, \mathsf{Copeland}\}$

Table 1: Summary of results

On the other hand, the new votes increase the Borda score of every candidate other than p and d by $2m^2h$ (relative to $\mathcal{R}'$), whereas the Borda score of p goes up by $2m^2h + 2mh$ and Borda score of d goes up by $2m^2h - 2mh$. Since the Borda scores of all candidates in $\mathcal{R}'$ do not exceed mh, p is the unique Borda winner in $\mathcal{R}''$ irrespective of how the manipulator votes. The rest of the proof is similar to that of Theorem 5.5 and Corollary 5.6. $\square$

We remark that the proofs of Theorems 4.3, 5.7 and 5.5 and Corollary 5.6 are based on the same idea: we can modify an election so that the (relative) scores of all candidates with respect to one rule remain essentially unchanged while making a certain candidate a winner with respect to another voting rule. This suggests that these rules exhibit certain independence; this is somewhat reminiscent of Klamler's work on closeness of voting rules (see Klamler [15] and references therein). Formalizing this notion of independence is an interesting direction for future work.

6. CONCLUSIONS AND FUTURE WORK

We have investigated the problem of (coalitional) manipulation under uncertainty about the voting rules. Our results are summarized in Table 1. While we have not established the complexity of our problem for all possible combinations of voting rules, our results identify a number of approaches for dealing with problems of this type and the features of voting rules that make their combinations easy or hard to manipulate.

An obvious direction for future work is extending our approach to other forms of cheating in elections, such as control and bribery. Also, an interesting variant of our problem in the context of single-winner manipulation can be obtained by adopting the paradigm of safe strategic voting [19]. That is, instead of assuming that the manipulator wants to get a certain candidate elected, we take the more traditional approach, where the manipulator, too, has a preference order and would like to improve the election outcome with respect to this order; we can then ask whether the manipulator can vote so that the outcome improves for at least one voting rule in the given family and does not get worse with respect to the other rules.

Acknowledgments This research was supported by National Research Foundation (Singapore) under grant 2009-08. We would like to thank the anonymous AAMAS referees for their very useful feedback.

7. REFERENCES

[1] Y. Bachrach, N. Betzler, and P. Faliszewski. Probabilistic possible winner determination. In *AAAI'10*, pages 697–702, 2010.

[2] J. Bartholdi and J. Orlin. Single transferable vote resists strategic voting. *Social Choice and Welfare*, 8(4):341–354, 1991.

[3] J. Bartholdi, C. Tovey, and M. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6(3):227–241, 1989.

[4] J. Bartholdi, C. Tovey, and M. Trick. How hard is it to control an election? *Mathematical and Computer Modeling*, 16(8/9):27–40, 1992.

[5] D. Baumeister, M. Roos, and J. Rothe. Computational complexity of two variants of the possible winner problem. In *AAMAS'11*, pages 853–860, 2011.

[6] N. Betzler, R. Niedermeier, and G. J. Woeginger. Unweighted coalitional manipulation under the Borda rule is NP-hard. In *IJCAI'11*, pages 55–60, 2011.

[7] V. Conitzer, T. Walsh, and L. Xia. Dominating manipulations in voting with partial information. In *AAAI'11*, pages 638–643, 2011.

[8] J. Davies, G. Katsirelos, N. Narodytska, and T. Walsh. Complexity of and algorithms for Borda manipulation. In *AAAI'11*, pages 657–662, 2011.

[9] E. Elkind, P. Faliszewski, and A. Slinko. Cloning in elections. In *AAAI'10*, pages 768–773, 2010.

[10] E. Elkind and H. Lipmaa. Hybrid voting protocols and hardness of manipulation. In *ISAAC'05*, pages 206–215, 2005.

[11] P. Faliszewski, E. Hemaspaandra, and H. Schnoor. Copeland voting: Ties matter. In *AAMAS'08*, pages 983–990, 2008.

[12] P. Faliszewski, E. Hemaspaandra, and H. Schnoor. Manipulation of Copeland elections. In *AAMAS'10*, pages 367–374, 2010.

[13] P. Faliszewski and A. D. Procaccia. AI's war on manipulation: Are we winning? *AI Magazine*, 31(4):53–64, 2010.

[14] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Hybrid elections broaden complexity-theoretic resistance to control. *Mathematical Logic Quarterly*, 55(4):397–424, 2009.

[15] C. Klamler. On the closeness aspect of three voting rules: Borda—Copeland—Maximin. *Group Decision and Negotiation*, 14:233–240, 2005.

[16] K. Konczak and J. Lang. Voting procedures with incomplete preferences. In *MPREF'05*, pages 124–129, 2005.

[17] J. Lang, 2010. Private communication.

[18] A. Lin. The complexity of manipulating k-approval elections. In *ICAART'11*, pages 212–218, 2011.

[19] A. Slinko and S. White. Non-dictatorial social choice rules are safely manipulable. In *COMSOC'08*, pages 403–413, 2008.

[20] L. Xia, V. Conitzer, and A. Procaccia. A scheduling approach to coalitional manipulation. In *ACM EC'10*, pages 275–284, 2010.

[21] L. Xia, M. Zuckerman, A. Procaccia, V. Conitzer, and J. Rosenschein. Complexity of unweighted manipulation under some common voting rules. In *IJCAI'09*, pages 348–353, 2009.

Voter Response to Iterated Poll Information

Annemieke Reijngoud and Ulle Endriss
Institute for Logic, Language and Computation (ILLC)
University of Amsterdam
Email: annemieke@averechts.nl, ulle.endriss@uva.nl

ABSTRACT

We develop a formal model of opinion polls in elections and study how they influence the voting behaviour of the participating agents, and thereby election outcomes. This approach is particularly relevant to the study of collective decision making by means of voting in multiagent systems, where it is reasonable to assume that we can precisely model the amount of information available to agents and where agents can be expected to follow relatively simple rules when adjusting their behaviour in response to polls. We analyse two settings, one where a single agent strategises in view of a single poll, and one where multiple agents repeatedly update their voting intentions in view of a sequence of polls. In the single-poll setting we vary the amount of information a poll provides and examine, for different voting rules, when an agent starts and stops having an incentive to manipulate the election. In the repeated-poll setting, using both analytical and experimental methods, we study how the properties of different voting rules are affected under different sets of assumptions on how agents will respond to poll information. Together, our results clarify under which circumstances sharing information via opinion polls can improve the quality of election outcomes and under which circumstances it may have negative effects, due to the increased opportunities for manipulation it provides.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*; J.4 [**Social and Behavioral Sciences**]: Economics

General Terms

Theory, Economics

Keywords

Computational Social Choice, Voting Theory

1. INTRODUCTION

Voting theory has recently come to play an important role in the study of multiagent systems [2, 12]. One of the most intriguing questions in voting is how agents strategise when casting their vote, in view of both their personal preferences and their beliefs about the strategies followed by others. In political elections, voters will often form their beliefs about the strategies of others on the basis of *opinion polls*. In this paper, we start from the same idea and propose a simple formal model for representing relevant information in a poll and the ways in which agents may respond to that information when deciding on what strategy to follow.

Most political elections are based on the *plurality rule*, under which a candidate obtains a point for every voter ranking her first, and the candidate with the largest number of points gets elected. In the context of this rule, the most natural type of information to publish in an opinion poll is the expected number of points for each candidate. Despite its widespread use, the plurality rule has been severely criticised for being overly simplistic and not allowing voters to adequately express their preferences. In multiagent systems, we have the opportunity to instead work with the full range of voting rules that have been proposed and studied in social choice theory [11]. This widening of the scope as far as the voting rule is concerned suggests to also consider a generalisation of the concept of opinion poll. To clarify this point, consider, for instance, the *Copeland rule*, under which you vote by submitting a strict linear order over the candidates and the score of a candidate is computed as the difference between the number of opponents she will beat in a one-to-one majority contest and the number of opponents she will lose to in such a contest (the candidate with the maximal Copeland score wins). In a poll, we could publish the (expected) Copeland score of each candidate or we could record how many copies of each possible ballot (linear order) were received. Alternatively, we could publish the majority graph (the directed graph on the set of candidates in which we include an edge from x to y if a majority of voters prefer x over y) or the weighted majority graph (in which each edge is annotated with the strength of the relevant majority).

To formally capture these ideas, we shall define a *poll information function* as a function mapping the ballots received from the voters in an opinion poll to an appropriate structure (e.g., a majority graph or a list of scores). The output of this function is then communicated to the voters, thereby providing them with partial information on the voting intentions declared by the others.

We shall analyse two scenarios. In the first, we study the incentives of a voter, who is provided with the output of a poll information function, to vote truthfully in a subsequent election. Intuitively, if a poll provides a lot of information,

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

then this will increase the opportunities of our voter to benefit from voting untruthfully, while a poll carrying very little information might be expected to reduce such opportunities and thereby induce truthful voting. Our results clarify, for several voting rules and several types of opinion polls, to what extent this basic intuition is in fact correct.

In the second scenario, voters participate in a sequence of polls, and in each round one of the voters can update her ballot in view of the poll information received. We consider several types of policies that a voter might use to perform this update: a *strategist* will choose a ballot such that no other ballot provides a strictly better outcome for some and at least as good an outcome for all possible ways of the other voters voting that would be consistent with the poll information received; a *pragmatist* will support her favourite candidate from a small set of, say, two front-runners; and a *truth-teller* will always vote truthfully. For different voting rules and for different combinations of these policies, we analyse whether such a system will converge to a stable state, i.e., whether it will terminate. We then observe that the "rule" we obtain when this kind of game is played for a specific voting rule and a specific set of response policies may be considered a voting rule in its own right, and we study how the properties of the original voting rule relate to the properties of this induced rule. An example for such a property is the frequency of electing a *Condorcet winner*, i.e., a candidate that would beat any other candidate in a one-to-one majority contest.

Similar phenomena have been studied before. Brams and Fishburn [1] give several examples that show, for both the plurality rule and another system known as *approval voting*, that executing a series of polls before the actual election can have both positive and negative effects in view of electing a Condorcet winner. Chopra et al. [4] give further examples, showing that a sequence of polls may or may not terminate. Meir et al. [7] identify conditions, in case the plurality rule is used, under which termination can be guaranteed. Finally, the work of Conitzer et al. [5] on the problem of strategic manipulation under partial information is closely related to the first scenario we study: an opinion poll is one way to model the partial information available to a manipulator. For comparison, most research on opinion polls in political science, such as the work of Irwin and Van Holsteyn [6], typically focuses on other concerns, e.g., the question of how polls affect the *expectations* of voters regarding election outcomes.

We proceed as follows. In Section 2 we review basic concepts from voting theory and define our model. Our results on strategic manipulation under limited information as provided by an opinion poll are presented in Section 3, and our results on voter response to iterated poll information are summarised in Section 4. Section 5 concludes. In the interest of space, we omit the proofs of some of our results. These proofs, as well as additional results, may be found in the Master's thesis of the first author [10].

2. THE MODEL

In this section we introduce our model. We first recall relevant concepts from voting theory [11], and then define the central notion of *poll information function*.

2.1 Voting Theory

Let $\mathcal{N}$ be a finite set of n *voters* (or *agents*), and let $\mathcal{X}$ be a finite set of m *alternatives* (or *candidates*). Each voter i is endowed with a *preference order* $\succ_i$ on $\mathcal{X}$. To vote, each voter i submits a *ballot* b_i (which may or may not be identical to $\succ_i$, i.e., she may or may not vote truthfully). We adopt the standard assumption that both preferences and ballots are strict linear orders on $\mathcal{X}$. Let $\mathcal{L}(\mathcal{X})$ be the set of all such orders. A *profile* $\boldsymbol{b} = (b_1, \ldots, b_n) \in \mathcal{L}(\mathcal{X})^{\mathcal{N}}$ is a vector of ballots, one for each voter. A *voting rule* $F : \mathcal{L}(\mathcal{X})^{\mathcal{N}} \to 2^{\mathcal{X}} \setminus \{\emptyset\}$ is a function mapping ballot profiles to nonempty sets of alternatives, the election winners. Most natural voting rules may produce ties, and thus a set of winners rather than a single winner. We can obtain a *resolute* voting rule, i.e., a rule that always returns a single winner, by pairing our voting rule of choice with a *tie-breaking rule*. We restrict attention to tie-breaking rules under which ties are broken according to some fixed but arbitrary order over the alternatives. W.l.o.g., we shall assume that this fixed order is the lexicographic order $a \succ b \succ c \succ \cdots$, i.e., we shall work with the *lexicographic tie-breaking rule*.

The following are examples for common voting rules [11]:

- *Positional scoring rules:* A PSR is defined by a scoring vector $(s_1, \ldots, s_m)$ with $s_1 \geqslant \ldots \geqslant s_m$ and $s_1 > s_m$. An alternative receives s_j points for each voter who ranks it at the jth position. The alternative(s) with the most points win(s) the election. Important PSRs are *plurality* with scoring vector $(1, 0, \ldots, 0)$, *antiplurality* (or *veto*) with scoring vector $(1, \ldots, 1, 0)$, and *Borda* with scoring vector $(m-1, m-2, \ldots, 0)$.
- *Copeland:* An alternative's score is the number of pairwise majority contests it wins minus the number it loses. The alternative(s) with the highest score win(s). A *pairwise majority contest* between alternatives x and y is won by x if a majority of voters ranks x above y.
- *Maximin* (also known as *Simpson*): An alternative's score is the lowest number of voters preferring it in any pairwise contest. The alternative(s) with the highest score win(s).
- *Bucklin:* An alternative's score is the smallest k such that a majority of voters rank the alternative in their top k. The alternative(s) with the lowest score win(s).
- *Single transferable vote:* An STV election proceeds in rounds. In each round the alternative(s) ranked first by the fewest voters get(s) eliminated. This process is repeated until only one alternative remains (or until all remaining alternatives are ranked first equally often).

Voting rules can be categorised by their formal properties, often referred to as *axioms* [11]. A voting rule is *anonymous* if it treats all voters symmetrically. A resolute voting rule is *surjective* if for every alternative there exists a profile under which that alternative wins. A *constant* voting rule is a rule that always elects the same unique winner. If there is a voter such that her top-ranked alternative is always the unique winner, then the voting rule is *dictatorial*. Otherwise it is *nondictatorial*. A voting rule is *unanimous* if it elects (only) alternative x whenever x is ranked first by all voters. A voting rule satisfies the *Pareto condition* if it does not return an alternative y that is ranked below some other alternative x by all voters. Note that any Pareto efficient rule is also unanimous (but not *vice versa*).

Finally, a voting rule is *Condorcet-consistent* if it elects (only) the Condorcet winner whenever it exists, and it is *strongly Condorcet-consistent* if it elects (only) the full set of weak Condorcet winners whenever that set is nonempty.

A *weak Condorcet winner* is an alternative that does not lose any pairwise majority contest, although it may tie some. A *Condorcet winner* wins any pairwise majority contest. Note that (weak) Condorcet winners only exist for some profiles. If a Condorcet winner does exist, then it must be unique, while there can be several weak Condorcet winners.

2.2 Polls and Poll Information Functions

In our model of an *opinion poll*, each voter is asked for her ballot.[1] We call the resulting ballot profile a *poll profile*. Often we would not want to communicate the whole poll profile to the electorate, e.g., to respect the privacy of voters or because it would be computationally too expensive to do so. Let $\mathcal{I}$ be the set of all possible pieces of poll information that we might want to communicate to the electorate in view of a given poll profile. A *poll information function* (PIF) is a function $\pi : \mathcal{L}(\mathcal{X})^{\mathcal{N}} \to \mathcal{I}$ mapping poll profiles to elements of $\mathcal{I}$. Here are some natural choices for $\mathcal{I}$ and the corresponding PIF π:

- *Profile:* The profile-PIF simply returns the full input profile: $\pi(\boldsymbol{b}) = \boldsymbol{b}$.
- *Ballot:* The ballot-PIF returns a vector recording how often each ballot occurs in the input profile.
- *(Weighted) Majority Graph:* The (W)MG-PIF returns the (weighted) majority graph of the input profile. A *majority graph* is a directed graph in which each node represents an alternative. There is an edge (x, y) from x to y if x wins their pairwise majority contest. In a *weighted majority graph*, each edge (x, y) is labelled with the difference in number between voters ranking x above y and voters ranking y above x.
- *Score:* Given a voting rule F, the corresponding score-PIF returns for each alternative its score under the input profile according to F. F should assign points to each alternative for this PIF to be well-defined.
- *Rank:* Given a voting rule F, the corresponding rank-PIF returns the rank of each alternative under the input profile according to F. F should rank all alternatives for this PIF to be well-defined.
- *Winner:* Given a voting rule F, the corresponding winner-PIF returns the winning alternative(s) under the input profile according to F: $\pi(\boldsymbol{b}) = F(\boldsymbol{b})$.
- *Zero:* The zero-PIF does not provide any information, i.e., it simply returns a constant value: $\pi(\boldsymbol{b}) = 0$.

Upon receiving the signal $\pi(\boldsymbol{b})$, and assuming she knows how π is defined, what can voter i infer about the poll profile $\boldsymbol{b}$? Of course, she knows her own ballot b_i with certainty. So, what can she infer about the remainder of the profile, $\boldsymbol{b}_{-i}$? We call the set of (partial) profiles that voter i must consider possible in view of the information she holds after receiving $\pi(\boldsymbol{b})$ her *information set*. It is defined as follows:

$$\mathcal{W}_i^{\pi(\boldsymbol{b})} \quad := \quad \{\boldsymbol{c}_{-i} \in \mathcal{L}(\mathcal{X})^{\mathcal{N}\setminus\{i\}} \mid \pi(b_i, \boldsymbol{c}_{-i}) = \pi(\boldsymbol{b})\}$$

We may think of $\pi(\boldsymbol{b})$ as the actual world and of $\mathcal{W}_i^{\pi(\boldsymbol{b})}$ as the set of possible worlds that are consistent with i's knowledge in world $\pi(\boldsymbol{b})$. Indeed, $\mathcal{W}$ satisfies the basic properties we would expect from a knowledge operator:

- (REF) $\boldsymbol{b}_{-i} \in \mathcal{W}_i^{\pi(a_i, \boldsymbol{b}_{-i})}$
- (SYM) if $\boldsymbol{b}_{-i} \in \mathcal{W}_i^{\pi(a_i, \boldsymbol{c}_{-i})}$, then $\boldsymbol{c}_{-i} \in \mathcal{W}_i^{\pi(a_i, \boldsymbol{b}_{-i})}$
- (TRA) if $\boldsymbol{b}_{-i} \in \mathcal{W}_i^{\pi(a_i, \boldsymbol{c}_{-i})}$ and $\boldsymbol{c}_{-i} \in \mathcal{W}_i^{\pi(a_i, \boldsymbol{d}_{-i})}$, then $\boldsymbol{b}_{-i} \in \mathcal{W}_i^{\pi(a_i, \boldsymbol{d}_{-i})}$

Axiom (REF) simply states that the actual poll profile is always part of every voter's information set. Axioms (SYM) and (TRA) together express that whenever a voter considers some ballot profile possible, then that profile would also induce her current information set. For a discussion of the knowledge-theoretic properties of polls in view of strategic voting we refer to the work of Chopra et al. [4].

We define the degree of "informativeness" of a PIF in terms of the information sets it induces:

Definition 1. *A PIF π is said to be* ***at least as informative*** *as another PIF σ, if $\mathcal{W}_i^{\pi(\boldsymbol{b})} \subseteq \mathcal{W}_i^{\sigma(\boldsymbol{b})}$ for all poll profiles $\boldsymbol{b} \in \mathcal{L}(\mathcal{X})^{\mathcal{N}}$ and all voters $i \in \mathcal{N}$.*

We note that Conitzer et al. [5] work with a similar notion of information set, except that they do not require an information set to be induced by poll information, but rather permit any set of conceivable profiles to form the information set of a given voter. Moreover, they do not include a voter's own ballot in her information set. There are also interesting connections to the work of Chevaleyre et al. [3] on the compilation complexity of voting rules: their *compilation functions* are the same types of functions as our PIFs.

3. RESPONSE TO A SINGLE POLL

In this section we analyse the case where, on the basis of an opinion poll, a single voter chooses to vote strategically.

3.1 Manipulation wrt. Poll Information

Suppose we run an opinion poll and communicate the result to voter i using PIF π (and suppose i did vote truthfully in the poll) and we then run the actual election. Will i have an incentive to manipulate and vote untruthfully, assuming she believes that the other voters will not change their ballot? We assume she does, if there is a scenario consistent with the poll information she received that will result in a better election outcome for her and there is no scenario where she will do worse than when voting truthfully.

Definition 2. *Let π be a PIF. Given a resolute voting rule F, a voter i, and a profile $\boldsymbol{b}$ with $b_i = \succ_i$, we say that i has an* ***incentive to π-manipulate*** *using ballot $c_i^\star$, if $F(c_i^\star, \boldsymbol{c}_{-i}) \succ_i F(\succ_i, \boldsymbol{c}_{-i})$ for some profile $\boldsymbol{c}_{-i} \in \mathcal{W}_i^{\pi(\boldsymbol{b})}$ and $F(c_i^\star, \boldsymbol{c}_{-i}) \succeq_i F(\succ_i, \boldsymbol{c}_{-i})$ for all other profiles $\boldsymbol{c}_{-i} \in \mathcal{W}_i^{\pi(\boldsymbol{b})}$.*

In above definition, $\succeq_i$ is the reflexive closure of $\succ_i$ and both relations are extended from elements of $\mathcal{X}$ to singleton sets over $\mathcal{X}$ in the natural manner. $F(\succ_i, \boldsymbol{c}_{-i})$ denotes the winning singleton under voting rule F when everyone votes as in profile $\boldsymbol{c}$, while voter i votes according to $\succ_i$, etc.

A voting rule is *susceptible to π-manipulation* if there is a voter who has an incentive to π-manipulate (for some $\boldsymbol{b}$ and some $c_i^\star$). If a resolute voting rule is not susceptible to π-manipulation, then it is *immune to π-manipulation.*

Lemma 1. *If a PIF π is at least as informative as another PIF σ, then any resolute voting rule that is susceptible to σ-manipulation is also susceptible to π-manipulation.*

[1]In most real-world opinion polls, pollsters do not ask *all* voters for their opinion. We can simulate this in our abstract model by simply dropping information on some of the voters before communicating the poll to the electorate.

PROOF. See Reijngoud [10]. $\square$

As an immediate corollary to Lemma 1, we obtain that, if π is at least as informative as σ, then any resolute voting rule that is immune to π-manipulation is also immune to σ-manipulation. In the sequel, we shall prove several susceptibility and immunity results for specific PIFs. Lemma 1 shows how such results can be transferred to other PIFs.

3.2 Susceptibility Results

When π is the profile-PIF, returning the full poll profile, then our notion π-manipulation reduces to the standard notion of manipulability used in voting theory. The seminal Gibbard-Satterthwaite Theorem [11] states that any resolute voting rule for three or more alternatives that is surjective and nondictatorial will be susceptible to manipulation. We shall now prove a simple generalisation of this theorem, relating the notion of π-manipulability applied and the informational requirements of the voting rule used.

Not every voting rule requires all information a ballot profile supplies to compute the winners. For the plurality rule, for example, it suffices to give for each alternative the number of ballots in which it is ranked first. For a given PIF $\pi : \mathcal{L}(\mathcal{X})^{\mathcal{N}} \to \mathcal{I}$, we say that a voting rule F is *computable from π-images* if there exists a function $H : \mathcal{I} \to 2^{\mathcal{X}}\backslash\{\emptyset\}$ such that $F = H \circ \pi$. We furthermore say that F is *strongly computable from π-images* if it is computable from π-images and $\pi(\boldsymbol{b}) = \pi(b_i, \boldsymbol{c}_{-i})$ entails $F(c_i, \boldsymbol{b}_{-i}) = F(\boldsymbol{c})$ for any two profiles $\boldsymbol{b}$ and $\boldsymbol{c}$, i.e., upon learning $\pi(\boldsymbol{b})$ a voter i can compute the winners for *any* way of voting herself (rather than just for b_i). For example, the Copeland rule is computable but not strongly computable from MG-information (i.e., from images under the MG-PIF), while it is strongly computable from WMG-information. Furthermore, any anonymous voting rule is strongly computable from ballot-information.

THEOREM 1. *Let π be a PIF. When $m \geqslant 3$, any resolute voting rule that is surjective, nondictatorial, and strongly computable from π-images is susceptible to π-manipulation.*

PROOF. Let π be a PIF with range $\mathcal{I}$ and let F be a resolute voting rule meeting above conditions. From the Gibbard-Satterthwaite Theorem it follows that F is susceptible to profile-manipulation, i.e., there exist a profile $\boldsymbol{b}$, a voter i, and a ballot $c_i^\star$ such that $F(c_i^\star, \boldsymbol{b}_{-i}) \succ_i F(\succ_i, \boldsymbol{b}_{-i})$. Since F cannot differentiate between profiles that produce the same $\mathcal{I}$-structure, we get $F(\succ_i, \boldsymbol{c}_{-i}) = F(\succ_i, \boldsymbol{b}_{-i})$ for any $\boldsymbol{c}_{-i}$ with $\pi(\succ_i, \boldsymbol{c}_{-i}) = \pi(\succ_i, \boldsymbol{b}_{-i})$. As F is *strongly* computable from π-images, this entails $F(c_i^\star, \boldsymbol{c}_{-i}) = F(c_i^\star, \boldsymbol{b}_{-i})$ for any $\boldsymbol{c}_{-i}$ with $\pi(\succ_i, \boldsymbol{c}_{-i}) = \pi(\succ_i, \boldsymbol{b}_{-i})$. Hence, F is susceptible to π-manipulation. $\square$

The conditions of Theorem 1 are not *necessary* for susceptibility. There are resolute voting rules that are surjective, nondictatorial, and susceptible to π-manipulation, yet not computable from π-images, as our next result shows.

THEOREM 2. *When $m \geqslant 3$ and n is even, any strongly Condorcet-consistent voting rule, paired with the lexicographic tie-breaking rule, is susceptible to MG-manipulation.*

PROOF. Let $\mathcal{X}$, $\mathcal{N}$ and F satisfy above conditions and let π be the MG-PIF. We construct a profile with three weak Condorcet winners such that voter i's second favourite alternative wins if she votes truthfully, and her first favourite wins if she votes untruthfully.

Fix $a, b, c \in \mathcal{X}$ with $a \neq b \neq c$. Let $\succ_i = b \succ a \succ c \succ \mathcal{X}\backslash\{a,b,c\}$, where alternatives $\mathcal{X}\backslash\{a,b,c\}$ are ranked in any order. And let $c_i^\star = b \succ c \succ a \succ \mathcal{X}\backslash\{a,b,c\}$. Let $\boldsymbol{b}_{-i}$ be a profile in which $\frac{n-2}{2}$ voters submit $b \succ a \succ c \succ \mathcal{X}\backslash\{a,b,c\}$, and $\frac{n-2}{2} + 1$ voters submit $c \succ a \succ b \succ \mathcal{X}\backslash\{a,b,c\}$. Then $F(\succ_i, \boldsymbol{b}_{-i}) = a$ (as ties are broken in favour of a) and $F(c_i^\star, \boldsymbol{b}_{-i}) = b$. It is not difficult to check that there is no profile $\boldsymbol{c}_{-i} \in \mathcal{W}_i^{\pi(\succ_i, \boldsymbol{b}_{-i})}$ such that $F(c_i^\star, \boldsymbol{c}_{-i}) \prec_i F(\succ_i, \boldsymbol{c}_{-i})$. It follows that F is susceptible to MG-manipulation. $\square$

Examples for voting rules that are strongly Condorcet-consistent include the maximin-rule, but not, for instance, the (Condorcet-consistent) Copeland rule.

Our final example for a π-susceptibility result concerns PSRs. Observe that a PSR is unanimous if and only if $s_1 > s_2$ holds for the scoring vector defining it.

THEOREM 3. *When $m \geqslant 3$ and $n \geqslant 4$, any unanimous positional scoring rule, paired with the lexicographic tie-breaking rule, is susceptible to winner-manipulation.*

PROOF. Let $\mathcal{X}$, $\mathcal{N}$ and F satisfy above conditions and let π be the winner-PIF wrt. F. We construct a profile where voter i's third favourite alternative wins if she votes truthfully and her second favourite wins if she votes untruthfully.

Fix $a, b, c \in \mathcal{X}$ with $a \neq b \neq c$. Let $\succ_i = c \succ a \succ b \succ \mathcal{X}\backslash\{a,b,c\}$, where alternatives $\mathcal{X}\backslash\{a,b,c\}$ are ranked in any order. And let $c_i^\star = a \succ c \succ b \succ \mathcal{X}\backslash\{a,b,c\}$. If n is odd, let $\boldsymbol{b}_{-i}$ be a profile in which $\frac{n-3}{2}$ voters submit $a \succ b \succ \mathcal{X}\backslash\{a,b\}$, $\frac{n-3}{2}$ voters submit $b \succ a \succ \mathcal{X}\backslash\{a,b\}$, and the remaining two voters submit $c \succ b \succ a \succ \mathcal{X}\backslash\{a,b,c\}$ and $b \succ a \succ c \succ \mathcal{X}\backslash\{a,b,c\}$. If n is even, let $\boldsymbol{b}_{-i}$ be a profile in which $\frac{n-2}{2}$ voters submit $a \succ b \succ \mathcal{X}\backslash\{a,b\}$, and $\frac{n-2}{2}$ voters submit $b \succ a \succ \mathcal{X}\backslash\{a,b\}$, and the remaining voter submits $b \succ c \succ a \succ \mathcal{X}\backslash\{a,b,c\}$. Since F is unanimous, i.e., $s_1 > s_2$, we get $F(\succ_i, \boldsymbol{b}_{-i}) = b$ and $F(c_i^\star, \boldsymbol{b}_{-i}) = a$. It is not difficult to check that there is no profile $\boldsymbol{c}_{-i} \in \mathcal{W}_i^{\pi(\succ_i, \boldsymbol{b}_{-i}))}$ such that $F(c_i^\star, \boldsymbol{c}_{-i}) \prec_i F(\succ_i, \boldsymbol{c}_{-i})$. It follows that F is susceptible to winner-manipulation. $\square$

3.3 Immunity Results

We now turn our attention to voting rules that are immune to certain types of manipulation. First, it is not difficult to verify that any *dictatorial* as well as any *constant* voting rule will be immune to *profile-manipulation* (and thus, by Lemma 1, also to any other form of manipulation). At the other extreme, as we shall see next, we can obtain immunity results for two large classes of voting rules with respect to the weakest form of immunity considered here, namely *zero-manipulation.* The next theorem is inspired by (and corrects a minor mistake in) a result due to Conitzer et al. [5].

THEOREM 4. *When $n \geqslant 3$, any strongly Condorcet-consistent voting rule, paired with the lexicographic tie-breaking rule, is immune to zero-manipulation.*

PROOF. Let $\mathcal{N}$ and F satisfy above conditions and let π be the zero-PIF. Fix any voter i, any poll profile $\boldsymbol{b}$ with $b_i = \succ_i$, and any untruthful ballot $c_i^\star$. Since $c_i^\star \neq \succ_i$, there is a pair of alternatives such that $x \succ_i y$ and $y \succ_{c_i^\star} x$. Claim: there exists a profile $\boldsymbol{c}_{-i}$ such that $F(c_i^\star, \boldsymbol{c}_{-i}) \prec_i F(\succ_i, \boldsymbol{c}_{-i})$. If n is odd, let $\boldsymbol{c}_{-i}$ be a profile in which $\frac{n-1}{2}$ voters submit $x \succ y \succ \mathcal{X}\backslash\{x,y\}$, and $\frac{n-1}{2}$ voters submit $y \succ x \succ \mathcal{X}\backslash\{x,y\}$, where alternatives $\mathcal{X}\backslash\{x,y\}$ are ranked in any order. If n is even, let $\boldsymbol{c}_{-i}$ be a profile in which $\frac{n-2}{2}$ voters submit $x \succ y \succ \mathcal{X}\backslash\{x,y\}$, and $\frac{n-2}{2}$ voters submit $y \succ x \succ \mathcal{X}\backslash\{x,y\}$, and the remaining voter submits $y \succ x \succ \mathcal{X}\backslash\{x,y\}$ in case x lexicographically precedes y and $x \succ y \succ \mathcal{X}\backslash\{x,y\}$ otherwise. Then $F(\succ_i, \boldsymbol{c}_{-i}) = x$ and $F(c_i^\star, \boldsymbol{c}_{-i}) = y$. Hence, for any untruthful ballot $c_i^\star$ there is a situation where i will do strictly better by voting truthfully. It follows that F is immune to zero-manipulation. □

Conitzer et al. [5] state a slightly stronger variant of Theorem 4: any resolute voting rule that is (not necessarily strongly) Condorcet-consistent is immune to zero-manipulation. This is true for an *odd* number of voters, as may be seen by revisiting the first part of our proof. For an *even* number of voters, however, Condorcet consistency is not sufficient, as demonstrated by the following example.

EXAMPLE 1. *Consider a scenario with 4 voters and 3 alternatives (a, b, c). Suppose that voting rule F elects the Condorcet winner if one exists, and otherwise the bottom choice of voter 1. Let $\succ_1 = a \succ b \succ c$, and consider ballot $c_1^\star = a \succ c \succ b$. Now there is a profile $\boldsymbol{c}_{-1}$ such that voter 1 benefits from voting untruthfully, namely when the others vote $a \succ b \succ c$, $b \succ a \succ c$, and $b \succ a \succ c$. Then $F(\succ_1, \boldsymbol{c}_{-1}) = c$ and $F(c_1^\star, \boldsymbol{c}_{-1}) = b$. It is not difficult to check that there is no profile $\boldsymbol{c}_{-1}$ such that $F(c_1^\star, \boldsymbol{c}_{-1}) \prec_1 F(\succ_1, \boldsymbol{c}_{-1})$. It follows that F is a resolute voting rule that is Condorcet-consistent and susceptible to zero-manipulation.*

We stress that Theorem 4 also cannot be simplified to stating that any voting rule that always elects *some* weak Condorcet winner whenever one exists is immune to zero-manipulation.

The following theorem strengthens another result by Conitzer et al. [5], who use a bound of $n \geqslant 6m - 12$.

THEOREM 5. *When $n \geqslant 2m - 2$, any positional scoring rule, paired with the lexicographic tie-breaking rule, is immune to zero-manipulation.*

PROOF. Let $\mathcal{N}$, $\mathcal{X}$ and F satisfy above conditions and let π be the zero-PIF. Fix any voter i and any poll profile $\boldsymbol{b}$ with $b_i = \succ_i$. And fix any untruthful ballot $c_i^\star$ such that $F(c_i^\star, \boldsymbol{c}_{-i}) \neq F(\succ_i, \boldsymbol{c}_{-i})$ for some profile $\boldsymbol{c}_{-i}$. Then there exists a pair of alternatives such that $x \succ_i y$ and $y \succ_{c_i^\star} x$, and x's score differs from y's in $\succ_i$ and $c_i^\star$. Claim: there exists a profile $\boldsymbol{c}_{-i}$ such that $F(c_i^\star, \boldsymbol{c}_{-i}) \prec_i F(\succ_i, \boldsymbol{c}_{-i})$. If n is odd, let $\boldsymbol{c}_{-i}$ be a profile in which $\frac{n-1}{2}$ voters submit $x \succ y \succ \mathcal{X}\backslash\{x,y\}$, and $\frac{n-1}{2}$ voters submit $y \succ x \succ \mathcal{X}\backslash\{x,y\}$, where every alternative $z \in \mathcal{X}\backslash\{x,y\}$ is ranked last by at least one voter. If n is even, let $\boldsymbol{c}_{-i}$ be a profile in which $\frac{n-2}{2}$ voters submit $x \succ y \succ \mathcal{X}\backslash\{x,y\}$, and $\frac{n-2}{2}$ voters submit $y \succ x \succ \mathcal{X}\backslash\{x,y\}$, where every alternative $z \in \mathcal{X}\backslash\{x,y\}$ is ranked last by at least two voters, and the remaining voter submits the ranking that is like $\succ_i$ but with x and y swapped in case x lexicographically precedes y and $c_i^\star$ with x and y swapped otherwise. Then $F(\succ_i, \boldsymbol{c}_{-i}) = x$ and $F(c_i^\star, \boldsymbol{c}_{-i}) = y$. Hence, F is immune to zero-manipulation. Observe that the bound of $n \geqslant 2m - 2$ follows from our requirements on the number of voters ranking each $z \in \mathcal{X}\backslash\{x,y\}$ last. (The case with 4 voters and 3 alternatives must be checked separately.) □

Together, Theorem 4 and Theorem 5 cover a broad range of voting rules. In particular, as is well known [11], the classes of Condorcet-consistent rules and PSRs do not overlap.

So far, all our immunity results involved either trivial voting rules (dictatorships and constant rules) or the trivial information set (for zero-manipulation). While we should not expect many positive results between these two extremes, they are not impossible to obtain either:

THEOREM 6. *When $n \geqslant 2m - 2$, the antiplurality rule, paired with the lexicographic tie-breaking rule, is immune to winner-manipulation.*

PROOF. Let $\mathcal{N}$ and F satisfy above conditions and let π be the winner-PIF wrt. F. W.l.o.g., we may assume that voters only submit an alternative they wish to veto. Fix any voter i, any profile $\boldsymbol{b}$ with b_i is voter i's true least favourite alternative, and any ballot $c_i^\star \neq b_i$. Claim: voter i never has an incentive to π-manipulate. Suppose $m \geqslant 3$ and $F(\boldsymbol{b}) = w$. If $w = b_i$, then i cannot change the outcome. If $w = c_i^\star$, let $\boldsymbol{c}_{-i}$ be a profile in which $n-1$ voters veto some $x \in \mathcal{X}\backslash\{b_i, w\}$, and all alternatives $x \in \mathcal{X}\backslash\{b_i, w\}$ are vetoed by at least one voter. If $w \in \mathcal{X}\backslash\{b_i, c_i^\star\}$, let $\boldsymbol{c}_{-i}$ be a profile in which $n-2$ voters veto some $x \in \mathcal{X}\backslash\{b_i, w\}$, and all alternatives $x \in \mathcal{X}\backslash\{b_i, w\}$ are vetoed by at least two voters, and the remaining voter vetoes w in case w lexicographically precedes b_i and some alternative $x \in \mathcal{X}\backslash\{b_i, w\}$ otherwise. Then $F(b_i, \boldsymbol{c}_{-i}) = w$ and $F(c_i^\star, \boldsymbol{c}_{-i}) = b_i$. Hence, F is immune to winner-manipulation. (The case with 2 alternatives must be checked separately.) □

THEOREM 7. *When $n \geqslant 10$, the plurality rule, paired with the lexicographic tie-breaking rule, is immune to MG-manipulation.*

PROOF. See Reijngoud [10]. The main insight is that for $n \geqslant 10$ the majority graph does not give enough information for a voter to infer the identity of the plurality winner. □

4. REPEATED RESPONSE TO POLLS

In this section we study the case where voters repeatedly react to opinion polls. We assume that all voters will vote truthfully in an initial poll. Then, given the poll information communicated to the voters (using a PIF), one voter is given the opportunity to update her ballot. This process is repeated until no further voter wishes to update her ballot (or until a given maximum number of rounds has been reached). We then apply the voting rule to this final profile to determine the election winner.

An important parameter in this kind of *voting game* is given by the *response polices* that voters use to update their ballots. We shall first formulate several such policies, and then formally introduce the voting games considered here. As we shall see, any such game (the definition of which includes a voting rule F) induces a new voting rule F^t (where t is the number of rounds played), and we will analyse the properties of F^t in view of those of F.

4.1 Response Policies

In each round, a voter i who has the opportunity to update her ballot must decide what to do based on her true preference order $\succ_i$, her previously submitted ballot b_i, and her current information set $\mathcal{W}_i$. A *response policy* determines for each voter i a function $\delta_i : \mathcal{L}(\mathcal{X}) \times \mathcal{L}(\mathcal{X}) \times 2^{\mathcal{L}(\mathcal{X})^{\mathcal{N}\backslash\{i\}}} \to \mathcal{L}(\mathcal{X})$, mapping $(\succ_i, b_i, \mathcal{W}_i)$ to a new ballot b_i'. We shall work with the following policies:

- *Truth-teller:* A truth-teller always votes truthfully, i.e., $\delta_i(\succ_i, b_i, \mathcal{W}_i) = \succ_i$.
- *Strategist:* A strategist computes her best responses to a poll and uses (any) one of them. Only if her current ballot is amongst the best responses, she will always use it. If we restrict attention to the plurality rule and assume that polls give score-information, then this policy is similar to the policy used by Meir et al. [7].
- *Pragmatist:* A pragmatist cannot or does not want to compute her best response to a poll, e.g., because this takes too much effort. A k-pragmatist always moves her favourite amongst the k currently highest ranked alternatives to the first position in her ballot, without changing the relative ranking of the others. This policy is also described by Brams and Fishburn [1].

We assume any given voter will use the same policy throughout. Voters also have to decide how to vote in the first round. As mentioned above, we assume that they all choose to vote truthfully then. This is not unreasonable, given the immunity results to zero-manipulation in Section 3.3.

Note that in the framework defined here, voters only take into account information from the latest poll round. However, the framework could be extended to include previous rounds by adding the information sets induced by those rounds to the input arguments of δ_i.

4.2 Induced Voting Rules

Let us now formally define the notion of *voting game*.

Definition 3. *A **voting game** is a tuple $G = \langle F, \pi, \boldsymbol{\delta} \rangle$, where F is a resolute voting rule, π is a PIF, and $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_n)$ is a vector of response policies.*

A voting game proceeds in rounds. Initially, all voters vote truthfully. In each subsequent round, exactly one voter changes her ballot. This voter is selected from the set of voters who wish to change. Whether or not a voter i wishes to do so depends on her response policy δ_i. At the end of each round, $\pi(\boldsymbol{b})$ is computed for the new poll profile $\boldsymbol{b}$ and the result is communicated to all voters. For a voting game to be uniquely defined, we need to fix the order in which voters may change their ballot. Any such order is allowed (none of our results will depend on the order chosen).

A voting game G *induces* a new voting rule F^t when the number of rounds to be played is t.

Definition 4. *Let $G = \langle F, \pi, \boldsymbol{\delta} \rangle$ be a voting game, and let $t \in \mathbb{N}$ be the number of rounds to be played. Then a voting rule F^t is **induced** by G by stipulating for any profile $\boldsymbol{b}$:*

$$F^t(\boldsymbol{b}) := \left\{ x \in \mathcal{X} \;\middle|\; \begin{array}{l} x \text{ is an election winner after } t \text{ rounds} \\ \text{when } \boldsymbol{b} \text{ is the truthful, initial profile} \end{array} \right\}$$

A game *terminates* in round t if no voter wishes to change her ballot according to her response policy at the end of t. If G always terminates after at most t rounds, then we write $F^\star$ instead of F^t. Clearly, if all voters are truth-tellers, then any voting game terminates after 0 rounds. Moreover, if F is immune to π-manipulation and all voters are strategists, then G terminates after 0 rounds as well.

Meir et al. [7] show that for any voting game G with F being the plurality rule and π the score-PIF wrt. the plurality rule, if p voters are strategists and $n{-}p$ voters are truth-tellers, then G terminates after at most $m \cdot p$ rounds. To be precise, these authors show this for a specific kind of strategist response policy, namely one in which a voter, whenever her current ballot is not amongst the best responses, changes her ballot to the best response in which the next alternative to win is the one she ranks first. We obtain a similar result for an electorate composed of truth-tellers and pragmatists (as opposed to strategists). In fact, under these assumptions the result can be generalised to arbitrary PSRs:

Lemma 2. *Let $G = \langle F, \pi, \boldsymbol{\delta} \rangle$ be any voting game such that F is a PSR, paired with the lexicographic tie-breaking rule, π is the rank-PIF wrt. F, and $\boldsymbol{\delta}$ is a vector of p k-pragmatists and $n{-}p$ truth-tellers. Then G terminates after $\leqslant p$ rounds.*

Proof. Let G, F, π, and $\boldsymbol{\delta}$ satisfy above conditions. Fix any profile $\boldsymbol{b}$. Let $H_k^t(\boldsymbol{b})$ be the set of k highest ranked alternatives in $\boldsymbol{b}$ according to F^t. Claim: $H_k^t(\boldsymbol{b}) = H_k^{t+1}(\boldsymbol{b})$ for any number of rounds $t \in \mathbb{N}$. Suppose that voter i changes her ballot at round t. Since i is a k-pragmatist, and F is a PSR, we have that no alternative $x \in H_k^t(\boldsymbol{b})$ loses points and no alternative $y \in \mathcal{X} \backslash H_k^t(\boldsymbol{b})$ wins any. Hence, each voter will update her ballot at most once and G terminates after at most p rounds. $\square$

A similar argument can be used to prove that if F is the Copeland rule and all voters are k-pragmatists, then G terminates after at most n rounds. This also holds in case F is the maximin rule or the Bucklin rule. On the other hand, games defined in terms of other voting rules or other response policies need not always terminate:

Example 2. *Consider a scenario with 2 voters and 3 alternatives (a, b, c). Let F be the Copeland rule, paired with the lexicographic tie-breaking rule. Let π be the MG-PIF. Suppose all voters are strategists. Consider the ballot profile $\boldsymbol{b} = (a \succ b \succ c,\ c \succ b \succ a)$. Then $F^0(\boldsymbol{b}) = a$, $F^1(\boldsymbol{b}) = b$, $F^2(\boldsymbol{b}) = a$, $F^3(\boldsymbol{b}) = c$, $F^4(\boldsymbol{b}) = a$, …(voters 2 and 1 alternate moving alternative b up and down in their ballots).*

4.3 Properties of Induced Voting Rules

For a given voting rule F, and under certain assumptions on the PIF used and the response policies of voters, what will be the properties of F^t (and $F^\star$, when it is well-defined)? This is the question we shall investigate next. Specifically, we are interested in properties that *transfer* from F to F^t and $F^\star$. Let us begin with a simple observation: If F is dictatorial, then any induced rule (for any PIF and any response policy defined here) will be dictatorial as well. More interestingly, we also obtain a transfer result for unanimity:

Theorem 8. *Let $G = \langle F, \pi, \boldsymbol{\delta} \rangle$ be any voting game such that F is unanimous, π is a PIF, and $\boldsymbol{\delta}$ is a vector of pragmatists, strategists and truth-tellers. Then F^t is unanimous for any $t \in \mathbb{N}$.*

Proof. Let G, F, π, and $\boldsymbol{\delta}$ satisfy above conditions. Fix any profile $\boldsymbol{b}$ such that there is an alternative w that is ranked first by all voters. Claim: $F^t(\boldsymbol{b}) = w$ for any number of rounds $t \in \mathbb{N}$. Proof by induction. Since F is unanimous, we have that $F^0(\boldsymbol{b}) = w$. Now, suppose that $F^t(\boldsymbol{b}) = w$, and that voter i wishes to change her ballot and may do so next. As no truth-teller or pragmatist who already has her favourite alternative winning will ever change her ballot, we only need to consider the case where i is a strategist. Since strategists always switch to a ballot that is at least as good as their previous ballot for all profiles in their information set (and strictly better for some), we have that $F^{t+1}(\boldsymbol{b}) = w$. This proves that F^t is unanimous for any $t \in \mathbb{N}$. □

However, the Pareto condition, which is slightly stronger than unanimity, does not always transfer:

EXAMPLE 3. *Consider a scenario with 2 voters and 3 alternatives (a, b, c). Let F be a voting rule that returns all alternatives that are not Pareto dominated, paired with the lexicographic tie-breaking rule. Let π be the winner-PIF wrt. F. Suppose all voters are strategists and let $t \geqslant 1$. Consider the ballot profile $\boldsymbol{b} = (b \succ c \succ a,\ c \succ a \succ b)$. Then $F^t(\boldsymbol{b}) = a$ (because the second voter will rank a on top, given that she has no chance to make c win, which is disadvantaged by the tie-breaking rule), even though alternative a is Pareto dominated by alternative c in profile $\boldsymbol{b}$.*

We also cannot guarantee the transfer of the Pareto condition for voting games in which all voters are pragmatists. Other properties that do not always transfer are surjectivity, anonymity, and Condorcet consistency. For all of these properties there are counterexamples in which all voters are strategists and polls give ballot-information. We omit these examples due to space constraints. On the other hand, under certain conditions, Condorcet consistency does transfer:

THEOREM 9. *Let $G = \langle F, \pi, \boldsymbol{\delta} \rangle$ be any voting game such that F is Condorcet-consistent, π is the rank-PIF wrt. F, and $\boldsymbol{\delta}$ is a vector of truth-tellers and pragmatists. Then F^t is Condorcet-consistent for any $t \in \mathbb{N}$.*

PROOF. Let G, F, π, and $\boldsymbol{\delta}$ satisfy above conditions. Fix any profile $\boldsymbol{b}$ with a Condorcet winner w. Claim: $F^t(\boldsymbol{b}) = w$ for any number of rounds $t \in \mathbb{N}$. Proof by induction. Since F is Condorcet-consistent, we have that $F^0(\boldsymbol{b}) = w$. Now, suppose that $F^t(\boldsymbol{b}) = w$, and that voter i wishes to change her ballot and may do so next. Since i is a k-pragmatist for some $k \in \mathbb{N}$, and w is among the k currently highest ranked alternatives, we have that w cannot lose support in any pairwise contest with respect to its original pairwise scores. It follows that $F^{t+1}(\boldsymbol{b}) = w$. This proves that F^t is Condorcet-consistent for any $t \in \mathbb{N}$. □

4.4 Condorcet Efficiency: Simulations

It is widely acknowledged that Condorcet consistency is a highly desirable property, but many important voting rules do not satisfy it [11]. The *Condorcet efficiency* of a voting rule is its tendency to elect the Condorcet winner. Theorem 9 identifies conditions under which Condorcet consistency transfers from F to F^t, but it does not say anything about the transfer of Condorcet efficiency. Brams and Fishburn [1] give several examples that show that polls can have a positive and a negative effect on the Condorcet efficiency of a voting rule. To study *how* positive or negative this effect is we shall make use of simulations.

We generated election data with n ranging from 10 to 100 in steps of 5 while keeping m fixed at 5, and m ranging from 3 to 15 in steps of 1 while keeping n fixed at 50 (using a simple program implemented in JAVA 1.6.0). For each of these combinations we generated 10,000 (truthful) ballot profiles with a Condorcet winner using the *impartial culture* (IC) assumption, which states that any permutation of alternatives is equally likely to occur as a voter's preference order. The limitations of the IC assumption are well known [9]; in particular, we should not expect the preferences in a real-world electorate to be distributed uniformly. Nevertheless, the IC assumption is still the *de facto* standard used in social choice theory; results based on it provide an important base line and allow for direct comparison with a large number of findings documented in the literature.

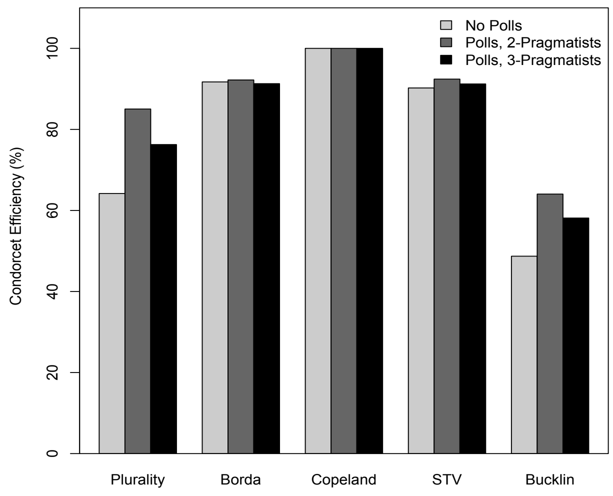


Figure 1: Average probability of electing the Condorcet winner for 50 voters and 5 alternatives over 10,000 trials. Poll effect on Condorcet efficiency is significant ($p < 0.05$) for plurality, STV and Bucklin.

Our first experiment was set up to test the effect of polls on the Condorcet efficiency of plurality, Borda, Copeland, STV, and Bucklin when polls provide (at least) rank-information, and all voters are 2-pragmatists or all voters are 3-pragmatists. That is, voters want to have as much electoral influence as they can without having to think too hard about a new ballot, unlike strategists. Note that we included the Copeland rule for comparison, but we already know that the rule itself is Condorcet-consistent, and thus its induced voting rule will be as well (cf. Theorem 9). We fixed the order in which voters may change their ballot to be the *ascending order*: voters were offered a chance to update their ballot according to their index in $\mathcal{N}$, beginning with the successor of the voter who was the last to change. All induced voting rules were run until termination.

Figure 1 shows the results for 50 voters and 5 alternatives; the other voter-alternative combinations showed a similar pattern, except for plurality, to which we will come back later. We used R to analyse our data [8], and McNemar's test to determine whether the poll effect was significant. The results for no-polls vs. polls for 2-pragmatists were $p = 0$, $p = 0.13$, $p < 0.001$, and $p = 0$ for plurality, Borda, STV, and Bucklin respectively. For no-polls vs. polls for 3-pragmatists they were $p < 0.001$, $p = 0.13$, $p < 0.001$, and $p < 0.001$ for the same rules. Thus, polls had a significant positive effect on the Condorcet efficiency of plurality, STV and Bucklin, and no significant effect for Borda.

Intuitively, one can think of the pragmatist response policy as offering a Condorcet winner another chance to win if it ended up among the k highest ranked alternatives in the first round. For plurality and Bucklin we can state this intuition as a general rule: if all voters are 2-pragmatists and the Condorcet winner is among the two highest ranked alternatives in the first round, then it will always win under induced vot-

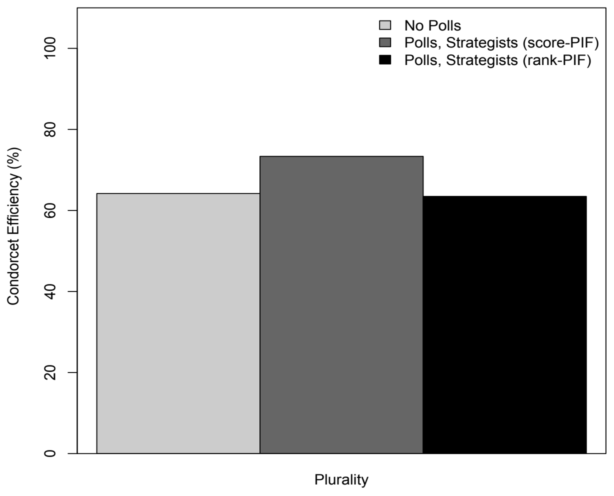


Figure 2: Average probability of electing the Condorcet winner for 50 voters and 5 alternatives over 10,000 trials. Poll effect on Condorcet efficiency is significant ($p < 0.05$) if polls give score-information.

ing rule $F^\star$. This follows from Lemma 2 and properties of F. However, if all voters are 3-pragmatists, then this no longer holds. Generally, an alternative in a runoff between 2 alternatives has a greater chance of winning to start with than an alternative in a runoff between 3. We would therefore expect the 2-pragmatist response policy to have a greater positive effect on the Condorcet efficiency than the 3-pragmatist response policy. Indeed, our data reflect this expectation. On the other hand, as m increases, it becomes less likely that the Condorcet winner ends up amongst the two highest ranked alternatives in the first round. This would explain that for large numbers of alternatives ($m \geqslant 12$), the 3-pragmatist response policy had a greater positive effect on the Condorcet efficiency of plurality than the 2-pragmatist response policy.

Our results also show that polls had a greater effect on plurality and Bucklin than on STV and Borda. This might be due to the substantially lower Condorcet efficiency of these rules, leaving more room for improvement.

So, polls improve the Condorcet efficiency of the widely used plurality rule if all voters are pragmatists. What about strategists? In our second experiment we tested the effect of polls on the Condorcet efficiency of plurality under the assumption that polls give rank- or score-information and that all voters are strategists. As under rank-information the induced rule did not always terminate, we ran $t = 10,000$ poll rounds, after which 38% of all elections did terminate. The results of the second experiment are shown in Figure 2. Again, we only show the results for elections with 50 voters and 5 alternatives. McNemar's test on paired results gave $p = 0.27$ for no-polls vs. rank-polls and $p < 0.001$ for no-polls vs. score-polls. Thus, polls had a significant positive effect on the Condorcet efficiency of plurality when voters received score-information. On the other hand, when polls only gave rank-information, we observed no significant effect on Condorcet efficiency. The latter effect, however, turned significantly positive for large m/n ratios, and significantly negative for small m/n ratios.

How can we explain this pattern? Providing a strategist with less information has two opposing effects: (1) she is more likely to update her ballot, because even if she actually cannot make a different alternative win, she may think she can; and (2) she is less likely to update her ballot, because she is risk-averse. Which effect is stronger is difficult to predict, but our simulation results suggest that this depends at least in part on the number of voters and alternatives.

5. CONCLUSION

We have developed a framework to study the effects of polls on voting behaviour and election outcomes. We found that when voters do not have any information about the voting intentions of others, then for many voting rules they never have an incentive to vote untruthfully. However, they start having these incentives as soon as they know who is currently winning, according to the poll. This does not necessarily mean that polls have a negative effect on the election outcome. Some favourable properties of voting rules do persist, and may even be strengthened, when a rule is complemented with a series of polls to which the voters can respond.

For properties that do not persist in general, further work on simulations, similar to our study of Condorcet efficiency, will be required. Beyond this, it would be interesting to investigate whether *combinations* of properties of voting rules induce particular properties of elections with polls. Finally, it would be worthwhile to consider additional types of policies according to which voters respond to poll information and to investigate their influence on election outcomes.

6. REFERENCES

[1] S. J. Brams and P. C. Fishburn. *Approval Voting.* Birkhäuser, 1983.
[2] Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. A short introduction to computational social choice. In *Proc. SOFSEM-2007*, 2007.
[3] Y. Chevaleyre, J. Lang, N. Maudet, and G. Ravailly-Abadie. Compiling the votes of a subelectorate. In *Proc. IJCAI-2009*, 2009.
[4] S. Chopra, E. Pacuit, and R. Parikh. Knowledge-theoretic properties of strategic voting. In *Proc. JELIA-2004*, 2004.
[5] V. Conitzer, T. Walsh, and L. Xia. Dominating manipulations in voting with partial information. In *Proc. AAAI-2011*, 2011.
[6] G. A. Irwin and J. J. M. van Holsteyn. According to the polls: The influence of opinion polls on expectations. *Pub. Opinion Quart.*, 66:92–104, 2002.
[7] R. Meir, M. Polukarov, J. S. Rosenschein, and N. R. Jennings. Convergence to equilibria in plurality voting. In *Proc. AAAI-2010*, 2010.
[8] R Development Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2011.
[9] M. Regenwetter, B. Grofman, A. A. J. Marley, and I. M. Tsetlin. *Behavioral Social Choice: Probabilistic Models, Statistical Inference, and Applications.* Cambridge University Press, 2006.
[10] A. Reijngoud. *Voter Response to Iterated Poll Information.* Master's thesis, ILLC, University of Amsterdam, 2011.
[11] A. D. Taylor. *Social Choice and the Mathematics of Manipulation.* Cambridge University Press, 2005.
[12] M. Wooldridge. *An Introduction to Multiagent Systems.* Wiley, 2nd edition, 2009.

Session 3D
Economies & Markets I

Rational Market Making with Probabilistic Knowledge

Abraham Othman
Computer Science Department
Carnegie Mellon University
aothman@cs.cmu.edu

Tuomas Sandholm
Computer Science Department
Carnegie Mellon University
sandholm@cs.cmu.edu

ABSTRACT

A market maker sets prices over time for wagers that pay out contingent on the future state of the world. The market maker has knowledge of the probability of realizing each state of the world, and of how the price of a bet affects the probability that traders will accept it. We compare the optimal policy for risk-neutral (expected utility maximizing) and Kelly criterion (expected $\log$-utility maximizing) market makers. Computing the optimal policy for a risk-neutral market maker is relatively simple, while computing the optimal policy for a Kelly criterion market maker is challenging, requiring advanced techniques adapted from the computational economics literature to run efficiently. We show that while a risk-neutral market maker has an optimal policy that does not depend on the market maker's state, a Kelly criterion market maker's optimal policy has an intricate dependence on both time and state. Counter-intuitively, a Kelly criterion market maker may offer bets that are myopically irrational with respect to the market maker's beliefs for the entire trading period. In contrast, a risk-neutral market maker never offers a myopically irrational bet.

Categories and Subject Descriptors

J.4 [**Social and Behavioral Sciences**]: Economics; I.2.11 [**Distributed Artificial Intelligence**]: Multi-agent Systems

General Terms

Algorithms, Economics, Experimentation

Keywords

Agent Design, Computational Economics, Interpolation, Numerical Dynamic Programming, Kelly Criterion, Wagering

1. INTRODUCTION

Market makers are trading agents that set the prices for assets in exchanges. Market makers profit in two ways: first, by the *bid/ask spread* imposed when they buy a contract at a lower price than they sell it for, and second, by speculatively taking on positions and holding that *inventory* for a profit. Many realistic market-making settings, like Las Vegas sports betting or a proprietary trading desk at a bank, are characterized by a market maker that has a good prior on the future state of the world and on how traders will bet as prices change. A complication arises when traders and the market maker have substantially different beliefs. Then, the market maker must balance two competing factors: the desire to hedge bets for a certain profit, and the desire to profit in expectation from wagers made at favorable prices. For instance, a market maker could find itself in a situation where it could either increase its exposure to an event it thinks will probably occur at a bargain price, or hedge out its current risk on that event in order to guarantee a small but certain profit.

In this paper, we compute the policy of a *Kelly criterion* market maker over a series of interactions with traders. The Kelly criterion [Kelly Jr, 1956] is a way to make bets that mandates maximizing the expected $\log$ utility of a setting. While simple as a guiding precept, the Kelly criterion accomplishes a broad range of objectives: over a series of bets, it is the fastest way to double an initial investment, produces the highest median wealth, and produces the highest mode wealth. Poundstone [2006] provides a compelling introduction to the Kelly criterion and its use in practice, particularly by the mathematician and hedge fund manager Ed Thorp.

There are two prior literatures that deal with the sequential interaction of a market maker with traders: artificial intelligence, and finance. In the AI literature, there are *cost function based market makers* [Chen and Pennock, 2007, 2010]—agents which price bets so that they are neutral between them being accepted and them being not accepted [Ben-Tal and Teboulle, 2007, Agrawal et al., 2009]. Another closely-related branch of the AI literature involves *Bayesian market makers* which attempt to learn the correct value of a security by applying Bayes Rule to a series of interactions with traders [Das, 2008, Das and Magdon-Ismail, 2009, Chakraborty et al., 2011]. In contrast to these agents, the market makers we consider here behave rationally in the classical sense: they maximize utility given knowledge of the future and of how their prices affect trader actions.

In the finance literature, Glosten and Milgrom [1985] defined the basic framework used in many later works, including ours, for a market maker interacting with an anonymous pool of traders. Kyle [1985] considers a game-theoretic interaction between a profit-seeking monopolistic market maker and a mix of noise traders and informed traders. Our setting is similar, in that we consider utility-maximizing pricing by a monopolistic market maker, but we assume that no traders are privileged in their information. The most challenging part of our work involves computing the policy of a risk-averse market maker. The notion of a risk-averse, rather than risk-neutral, market maker was introduced in Rock [1996].

Our experimental results show that a Kelly criterion market maker follows a complex time-dependent strategy. In the early stages of wagering, the market maker will attempt to match orders to profit from the bid/ask spread. Towards the end of trading, the policy

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

gradually shifts to myopic optimization on the market maker's private beliefs. Perhaps surprisingly, we show that in the early stages of the market, profiting from the bid/ask spread dominates the desire to sell inventory at agreeable prices, that is, if it facilitates more trade, a Kelly criterion market maker should buy obligations at a price higher than, or sell obligations at a price lower than, its private beliefs. Moreover, because the inventory a risk-averse market maker accumulates affects the prices it offers, the market maker could offer bets that are myopically irrational for the entire trading period. This is in contrast to a risk-neutral market maker that would never offer a myopically irrational bet.

2. MODEL

Following Glosten and Milgrom [1985], the setting is a repeated sequential interaction between the market maker and a set of traders. In each period, the market maker sets prices for a finite set of bets, and then a trader is drawn randomly from a large pool of potential traders. That trader enters the market and selects one of the offered bets to make with the market maker (or none at all). After a finite number of periods the process halts, one of the n events is realized, and the bets are settled with the traders.

Our setting is closest to Das and Magdon-Ismail [2009], which also involved a dynamic stochastic optimization. In that work, however, the market maker was responding to a shock in the state of the world and was attempting to learn the new, correct values for contracts. In contrast, here, the market maker's beliefs over the probabilities of the future state of the world do not change, and the market maker seeks to maximize expected utility over the interaction with the traders.

2.1 Traders

The traders have the following features:

- Traders are *anonymous*, so there is no way for the market maker to distinguish between traders. Anonymity is a standard component of many models in the literature (for example, Feigenbaum et al. [2003] and Das [2008]), because it is natural for settings where prices are posted publicly, as is the standard in electronic markets.

- Traders are *myopic*, not strategic. They exist for only a single period: they enter the market, perceive the prices offered by the market making agent, select a bet to take (or no bet), and then exit. The traders do not learn from historical prices or strategize about their behavior. Myopic traders (also known as *noise traders*) are a feature of much of the literature [Glosten and Milgrom, 1985, Kyle, 1985, Othman and Sandholm, 2010]. Empirical studies of market microstructure have shown that the behavior of these agents is qualitatively very similar to behavior observed in real markets with human traders [Gode and Sunder, 1993, Othman, 2008]. However, in some settings the simple behavior of these agents may be an unrealistic model [Chen et al., 2007, Dimitrov and Sami, 2008, Chen et al., 2010].

- The number of trading periods is drawn independently of the market maker's policy. Since traders have the ability to decline to place a bet with the market maker if they do not find the offered bets agreeable, this condition means that the number of traders placing bets with the market maker is *not* a constant—instead, it will depend on the market maker's policy. We assume the market maker knows the true distribution of the number of trading periods.

2.2 Utility and the Bellman equation

The market maker's *state* can be represented by a tuple $(t, \mathbf{w})$ of the index of the participating agent $t \in \{1, 2, \ldots\}$, and the *wealth vector* $\mathbf{w}$, where w_i is the market maker's wealth (payoff) if state of the world $\omega_i \in \Omega$ is realized. (Since exactly one trader appears in each period, the variable t can be thought of as an index over discrete time.) There is a termination state $(\bar{t}, \mathbf{w})$, where the market maker gets an expected utility payout based on his subjective beliefs $\hat{\mathbf{p}}$, which he believes to be the correct distribution over the possible futures:

$$V(\bar{t}, \mathbf{w}) \equiv \sum_{i=1}^{n} \hat{p}_i u(w_i)$$

Without loss of generality, a risk-neutral market maker receives its expected linear utility on termination:

$$V(\bar{t}, \mathbf{w}) \equiv \sum_{i=1}^{n} \hat{p}_i w_i$$

A Kelly criterion market maker receives its expected log utility on termination:

$$V(\bar{t}, \mathbf{w}) \equiv \sum_{i=1}^{n} \hat{p}_i \log(w_i)$$

The *bets* a market maker offers can be expressed by vectors in payout space $\mathbf{x} \in \mathbb{R}^n$, so that x_i is the *trader's* payoff (that is, the market maker's loss) if ω_i is realized. For instance, imagine that the market maker is fielding bets on which of three horses will win a horse race. A bet that pays the trader 10 dollars if the first horse wins, 5 dollars if the second horse wins, and nothing if the third horse wins, is represented by the vector $(10, 5, 0)$.

The market maker's *policy* when interacting with trader t, $\pi(t, \cdot) : \mathbb{R}^n \mapsto \mathbb{R}$, maps these vectors to the amount the market maker would charge the agent for each bet. We denote by the zero-vector bet $\mathbf{0}$ an agent declining to make a bet with the market maker, and set $\pi(\mathbf{0}) = 0$. (This can be interpreted as the intersection of the individual rationality constraint of the traders (who would want $\pi(\mathbf{0}) \leq 0$) and of the market maker (who would want $\pi(\mathbf{0}) \geq 0$).) The market maker knows the probability that an agent will accept a bet given the prices. Because traders are anonymous, the market maker has no way to distinguish between traders and so these probabilities are the same for all traders.

In full generality, there is a chance $\delta(t)$ of the interaction terminating immediately before the t-th trader participates. Consequently, the value of being in state $(t, \mathbf{w})$ is

$$\begin{aligned} V(t, \mathbf{w}) = (1 - \delta(t)) \sum_{\mathbf{x}} & \mathbb{P}\left(\text{Trader takes bet } \mathbf{x} \text{ at price } \pi(\mathbf{x})\right) \\ & \cdot V(t+1, \mathbf{w} - \mathbf{x} + \pi(\mathbf{x})) \\ + \delta(t) V(\bar{t}, \mathbf{w}) & \end{aligned}$$

In every state $(t, \mathbf{w})$, a utility-maximizing market maker employs the optimal policy π^* defined by the Bellman equation

$$\begin{aligned} \pi^*(t, \mathbf{w}) = \arg\max_{\pi} (1 - \delta(t)) \sum_{\mathbf{x}} & \mathbb{P}\left(\text{Trader takes bet } \mathbf{x} \text{ at price } \pi(\mathbf{x})\right) \\ & \cdot V(t+1, \mathbf{w} - \mathbf{x} + \pi(\mathbf{x})) \\ + \delta(t) V(\bar{t}, \mathbf{w}) & \end{aligned}$$

with respective values V^* defined by

$$V^*(t,\mathbf{w}) = (1-\delta(t))\sum_{\mathbf{x}}\mathbb{P}\left(\text{Trader takes bet } \mathbf{x} \text{ at price } \pi^*(\mathbf{x})\right) \cdot V(t+1, \mathbf{w}-\mathbf{x}+\pi^*(\mathbf{x})) + \delta(t)V(\bar{t},\mathbf{w})$$

Solving these equations when the market maker has log utility is very challenging. We proceed to discuss how we solve for the optimal policy and values in this case.

3. COMPUTATION OF THE POLICY OF A KELLY CRITERION MARKET MAKER

When given a specification of the value function $V^*(t+1,\mathbf{w})$, it is simple to calculate the optimal value V^* and policy π^* of any state in the previous time step t. Thus, backward induction from the termination state is a straightforward way to solve for optimal values and policy across every time step. A complication arises from the difficulty in representing arbitrary $V^*(t+1,\mathbf{w})$. While the termination state is closed form, the previous time steps will generally not have closed form representations. In order to solve a Kelly criterion market maker's problem with backward induction, we must find a way to approximately represent the value function concisely.

3.1 Shape-preserving interpolation

While the value function for an arbitrary time step may have a complex, non-analytic form, we know a great deal about its *shape* from the properties it inherits from the log utility of the terminating state [Stokey et al., 1989]. In particular: (1) it is increasing in wealth, (2) it is concave, and (3) it goes to minus infinity as the wealth in any state goes to zero.

Since these properties are intrinsically linked to the logarithmic utility of the Kelly criterion market maker, we choose to adopt an approximation technique that preserves these properties, *shape-preserving interpolation*. Specifically, we employ the shape-preserving interpolation developed theoretically in Constantini and Fontanella [1990]. By shape-preserving, we mean that the technique retains the partial derivatives, concavity, and monotonicity of the original function, and by interpolation, we mean that the approximated function precisely matches the actual function at a set of interpolating points. While shape-preserving interpolation is well-known in the scientific computing literature [Judd, 1998], this specific technique has been featured rarely. Perhaps the most practical example is Wang and Judd [2000], who study a tax planning problem with stochastic stocks and bonds.

Because the theory of shape-preserving interpolation developed in Constantini and Fontanella [1990] is complete only for two dimensions, we focus only on settings with two events for the rest of the paper. While it does appear possible to extend the interpolation into n dimensions, it would suffer from the curse of dimensionality and take significantly longer to compute the approximate value function. The restriction to two events is not as limiting as it might first appear, because many realistic and popular settings involve wagers on binary events. An example from sports betting is whether the Red Sox or Yankees will win their upcoming match. An example from finance is credit-default swaps, where a bond either does or does not experience a default event.

In order to properly preserve the shape of the function, shape-preserving interpolation requires computing the partial derivatives with respect to the wealth in each state at the interpolating points. We compute these values by using the *envelope theorem*; since $V^*(t,\mathbf{w})$ is given by the maximizing policy π^*, we calculate the partial derivatives with respect to wealth by numerically differentiating the value function when the maximizing policy is followed [Mas-Colell et al., 1995, Wang and Judd, 2000].

We proceed to describe the interpolation procedure at a high

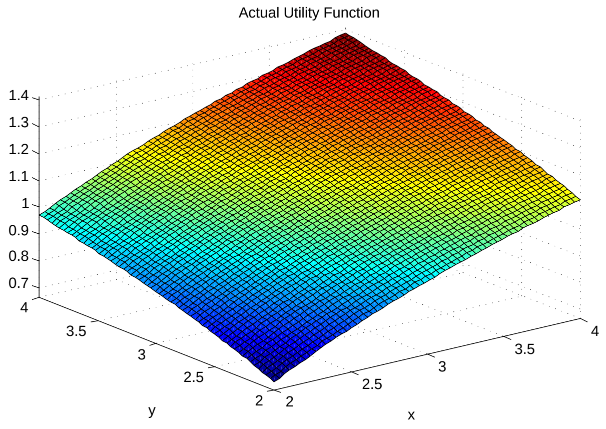


Figure 1: The utility function $u(x,y) = .6\log x + .4\log y$ on the rectangle $[2,4]^2$.

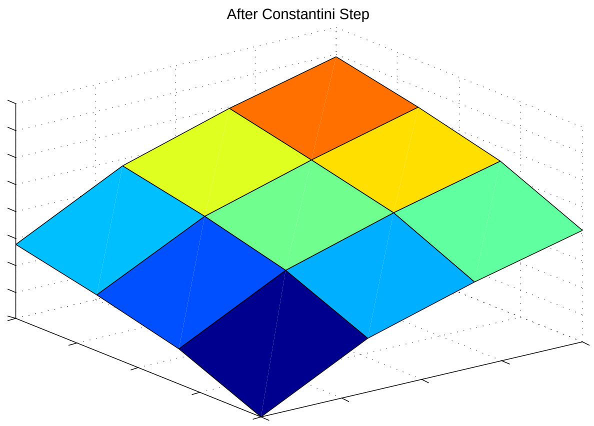


Figure 2: The quilt which matches the function values and partial derivatives.

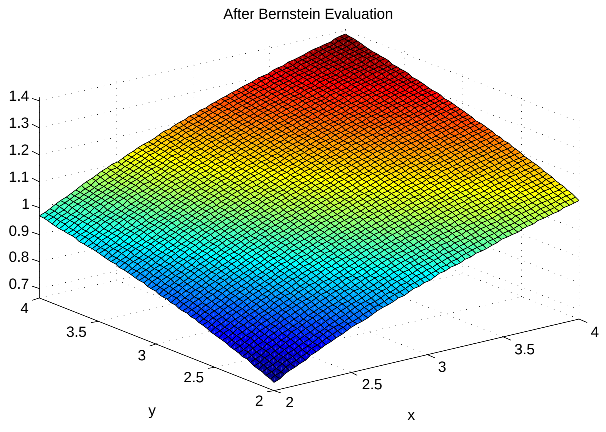


Figure 3: Evaluating the quilt using Bernstein bases produces a good approximation.

level, first on a single rectangle and then over the whole positive orthant. Figure 1 shows a sample expected utility function over a single rectangle.

The first step to creating an approximate interpolating function on this rectangle is to generate a three-by-three *quilt* (continuous, piecewise-linear approximation) of the function by matching the function values and partial derivatives at the vertices of the rectangle. Figure 2 shows the quilt that results from the utility function in Figure 1. This quilt retains the monotonicity, concavity, and partial derivatives at the vertices of the original function.

The final step is to evaluate the quilt using *bivariate Bernstein basis functions*. These are a variation-minimizing set of functions that retain the monotonicity, concavity, and partial derivatives at the vertices of the quilt. Of course, the quilt retained these properties from the original function itself, and so the interpolation is shape preserving. By variation minimizing, we mean that the bases are weighted to produce a polynomial that minimizes the sup ($\mathcal{L}_\infty$) norm error. It is therefore accurate to think of the Bernstein bases as smoothing the piecewise linear quilt [Judd, 1998]. Figure 3 shows the interpolated function that results from the process. Since the Bernstein evaluation step works directly on the quilt, the function is approximated concisely: for each interpolating rectangle we only need to store the sixteen values that create the quilt.

Computing the shape-preserving interpolated function is more involved than a simple linear interpolation (table lookup). However, the benefit of these extra steps is the dramatically improved accuracy of the evaluated function or, put another way, a substantial decrease in the degree of grid fineness required to compute the value function to the same level of accuracy. Table 1 compares the accuracy of the shape-preserving interpolation versus a simple linear interpolation at an arbitrary collection of wealth vectors for the representative utility function used in Figure 1.

Wealth vector	Shape-preserving error	Linear error	Ratio
$(2, 2)$	.0020	.14	72
$(5, 1.1)$	.0026	.36	135
$(20, 25)$	1.2×10^{-6}	.0011	895
$(50, 10)$	1.0×10^{-5}	.0021	210

Table 1: Relative errors for shape-preserving interpolation versus linear interpolation on identical rectangles. At each wealth vector, the interpolating rectangle is $(w_1 \pm 1, w_2 \pm 1)$, i.e., a square with side length 2 centered at the wealth vector.

The shape-preserving interpolation is between 72 and 895 times more accurate than a linear grid at the example points. Perhaps unsurprisingly, we found that the inverse of this relation also appeared to hold—to achieve the same level of accuracy as shape-preserving interpolation, the grid used in linear interpolation would need to be roughly one thousand times finer. We estimate that the running time of our experiments on a commodity PC using linear interpolation would take about a week; in contrast, solving the dynamic program took about ten minutes using shape-preserving interpolation.

3.2 Extending the technique

We have described how shape-preserving interpolation works on a single rectangle over which the function to be approximated is finite. It is straightforward to extend this technique from a single rectangle to a finite grid of rectangles over which the function to be approximated is finite. (In this case, care must to be taken to ensure that the function approximation is continuous at the boundaries of the individual interpolating rectangles, but this can be accommodated without too much additional complexity, see Constantini and Fontanella [1990] for details.)

However, the value function we are approximating is not just a finite function over a finite grid: it fails this in two separate ways. First, since $\lim_{x \downarrow 0} \log x = -\infty$, we have that at the lower boundary of the positive orthant (i.e., values close to zero along either dimension) the value function goes to $-\infty$. Second, the value function has no finite upper bound on its input—it is defined over the entire positive orthant. Consequently, we must extend the interpolation technique from the literature to accommodate the specific properties of a Kelly criterion market maker. Our solution is to have a large finite grid of interpolating rectangles on which we can apply the standard shape-preserving technique, and then to employ custom extensions to approximate below the lower boundary and above the upper boundary of the grid.

3.2.1 Beyond the lower boundary of the grid

We interpolate beyond the lower boundary of the grid as if the value were given by setting the value of a state equal to its termination value plus a constant that ensures continuity at the boundary of the grid. Formally, to approximate the value of state $\mathbf{w}$, with nearest point on the interpolating grid $\mathbf{w_g}$, we set

$$V(t, \mathbf{w}) \approx V(\bar{t}, \mathbf{w}) + (V(t, \mathbf{w_g}) - V(\bar{t}, \mathbf{w_g}))$$

(Observe that as $\mathbf{w} \to \mathbf{w_g}$, $V(t, \mathbf{w}) \to V(t, \mathbf{w_g})$). This approximation ensures the monotonicity of the value function and that it goes to negative infinity as the wealth of either state goes to zero, but, it is only an exact approximation for the termination function itself. To ensure that this extension does not change the overall value function substantially, in our experiments we start the interpolating grid at a small value, so the additional interpolation is only relevant over a small fraction of the state space. In our exploratory data analysis, we experimented with different lower bounds for the interpolating grid and found that different small values did not noticeably affect calculated optimal policies. We attribute this to states at the lower boundary of the grid having such low utility that they will be avoided, and are therefore largely irrelevant to the optimization problem as a whole.

3.2.2 Beyond the upper boundary of the grid

Consider the market maker's pricing problem at the upper boundary of the grid at time t. If the size of the trader's bet is bounded (say, to be no larger than c), then the market maker can approximately compute the optimal pricing policy by using an interpolating grid at time step $t + 1$ whose upper boundary is larger than the grid at time t by at least c. Using this insight, we eliminate the need to calculate a value beyond the upper boundary of the grid by increasing the upper boundary of the grid as time proceeds. (In fact, recalling that we solve the dynamic program through backward induction, from an algorithmic perspective we are actually reducing the upper boundary of the grid as we solve backwards through time.) In contrast to our extension to compute values below the lower boundary of the interpolating grid that we discussed above, this extension uses the same mechanics as the rest of the shape-preserving interpolation process and so suffers from no additional loss of accuracy.

3.3 Alternative approaches

As an alternative to the gridded approach here, we also considered but rejected a global shape-preserving approximation technique along the lines of De Farias and Van Roy [2003]. This would involve selecting basis functions ϕ_i that are each monotonic and concave, and representing the value function in each time step as a conical combination of these functions:

$$V^*(t, \mathbf{w}) \approx \sum_i \gamma_i^t \phi_i(\mathbf{w}), \quad \gamma_i^t \geq 0.$$

Such a representation retains the monotonicity and concavity properties of the value function and is concise. We rejected this approach for two reasons. First, the heuristic selection of the basis provides little guidance. Which set of functions is a good choice, and why? Observe that many standard basis function selections, such as radial basis functions, will not in general preserve the monotonicity or concavity of the value function and so could lead to nonsensical policies.

The second reason we chose to reject this technique is the difficult optimization to select the weights γ^t. In particular, a standard linear regression that maximizes the deviation from sum of squares at a set of relevant nodes can create aberrant behavior and an approximation that deviates significantly from the actual value function [Gordon, 1995, Guestrin et al., 2001, Stachurski, 2008]. The correct optimization to use to determine the weights is to minimize the sup norm (that is, $\mathcal{L}_\infty$, rather than $\mathcal{L}_2$), which is a significantly more challenging problem to solve numerically [Judd, 1998].

4. EXPERIMENTS

With only two possible events, it is possible to characterize bets in terms of a single event. In particular, setting p to be the probability that the first event occurs implies $1 - p$ is the probability that the second event occurs. Applying this logic to the market maker's policy, we can without loss of generality have the market maker buy and sell contracts on the first event only, because buying (selling) a contract on the first event implicitly yields the sale (purchase) of a contract on the second.

The *ask* is the price at which the market maker will sell a contract, and the *bid* is the price at which the market maker will buy a contract. For non-degenerate settings, ask prices will always be higher than bid prices. In this section, we describe the optimal ask and bid prices for two different settings.

4.1 Parameterization

Following Das [2008], in our experiments, traders have a belief drawn from a Gaussian with a mean belief of $p = 0.5$ and standard deviation 0.05. The traders are zero-intelligence agents; a trader visits the market maker exactly once and behaves myopically. They purchase a unit contract if they see an ask price lower than their belief, sell a unit contract if they see a bid price higher than their belief, and do not transact with the market maker otherwise.

In our experiments, we set 50 trading periods (that is, $\delta(t = 51) = 1, \delta(t < 50) = 0$), although we found our results hold qualitatively for other distributions of traders. Recalling from Section 3.2 that the upper boundary of the interpolating grid increases in each trading period, we set the interpolating grid for trader t to $[1, 1.5, 2, 3, \ldots, 250, 250 + t]^2$.

In our experiments with Kelly criterion market makers, we consider only relatively small levels of wealth (alternatively, large bets relative to the amount of wealth). This is because for bets with large levels of wealth, a market maker maximizing the expected log of wealth can be well-approximated by a risk-neutral, linear utility agent. To see why, consider the Taylor expansion of log utility at wealth x:

$$\log(x + \epsilon) = \log(x) + \frac{\epsilon}{x} - \frac{\epsilon^2}{x^2} + \Theta\left(\epsilon^3\right)$$

If x is large enough that $x^2 \gg x$, then $1/x^2 \ll 1/x$. Consequently, at large wealths, the impact of small bets on the utility function can be well-approximated by the linear function $\log(x) + \left(\frac{1}{x}\right)\epsilon$, with negligible higher-order effects.

We now turn our attention to how to calculate the optimal policy for a risk-neutral market maker, and the qualitative properties of that policy.

4.2 Optimal risk-neutral policy

For this setting, a risk-neutral market maker's optimization problem is significantly simpler than the general case. Recall that in the two-event case, the market maker's knowledge of the future, the vector $\hat{\mathbf{p}}$, can be represented by a single scalar $\hat{p}$ (e.g., "Team A has a 50 percent chance of winning the game"). Then the termination state $V(\bar{t}, \mathbf{w})$ is

$$V(\bar{t}, \mathbf{w}) \equiv \hat{p} w_1 + (1 - \hat{p}) w_2$$

Let the agents have beliefs on the first event distributed according to the cumulative density function F with probability density function f. In the penultimate step $\bar{t} - 1$, a risk-neutral market maker sets their bid and ask price to maximize their utility in the termination state, conditioning on three cases: the bid being taken, the ask being taken, and neither offer being taken. Formally,

$$\begin{aligned} V^*(\bar{t} - 1, \mathbf{w}) \quad = \quad & \max_{b,a} F(b) V(\bar{t}, (w_1 - b + 1, w_2 - b)) \\ & + (1 - F(a)) V(\bar{t}, (w_1 + a - 1, w_2 + a)) \\ & + (F(a) - F(b)) V(\bar{t}, \mathbf{w}) \end{aligned}$$

and since $V(\bar{t}, \mathbf{w}) = \hat{p} w_1 + (1 - \hat{p}) w_2$ the right-hand side optimization simplifies to

$$\begin{aligned} \max_{b,a} \quad & F(b)(\hat{p}(w_1 - b + 1) + (1 - \hat{p})(w_2 - b)) \\ & + (1 - F(a))(\hat{p}(w_1 + a - 1) + (1 - \hat{p})(w_2 + a)) \\ & + (F(a) - F(b))(\hat{p} w_1 + (1 - \hat{p}) w_2) \end{aligned}$$

which further simplifies to

$$\max_{b,a} V(\bar{t}, \mathbf{w}) + F(b)(\hat{p} - b) + (1 - F(a))(a - \hat{p}) \qquad (1)$$

which implies

$$V^*(\bar{t} - 1, \mathbf{w}) + C = V(\bar{t}, \mathbf{w})$$

where C is a constant that does not depend on t or $\mathbf{w}$. Consequently, by inductive argument working back from the terminal state the optimal policy for a risk-neutral market maker does not depend on t or $\mathbf{w}$. Equation 1 also makes it easy to see that the optimal arguments (b^*, a^*) have $b^* \leq \hat{p} \leq a^*$, because if not changing to a policy satisfying that inequality would yield a higher value. Thus, a globally optimal risk-neutral market maker is always myopically rational.

This argument also applies to the general setting discussed in Section 2.2 with more than two bets and events. In that case, by similar reasoning, the result is that a risk-neutral market maker will always price a bet $\mathbf{x}$ such that $\pi(\mathbf{x}) \geq \hat{\mathbf{p}} \cdot \mathbf{x}$. In this more-advanced case, however, traders' demands could be a complex, combinatorial function of the price vector offered by the market maker. If so, computing the optimal policy could be infeasible.

We have shown that the optimal policy of a risk-neutral market maker is constant and invariant to time and wealth. To actually

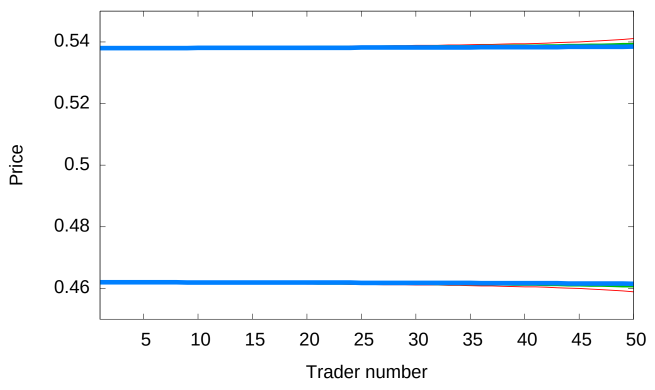


Figure 4: When the market maker's private beliefs align with those of the traders, the optimal ask prices (top lines) and bid prices (bottom lines) do not change significantly over the course of the interaction period.

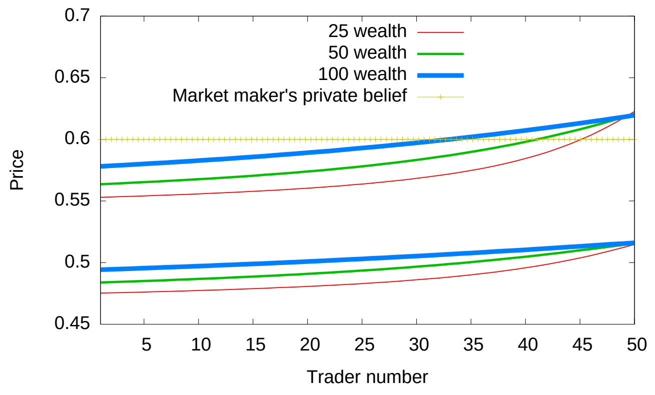


Figure 5: When the market maker's private beliefs do not align with those of the traders, the optimal policy is highly time and wealth dependent.

compute the optimal b^* and a^*, in the simple two-event, unit-bet case we can take the first-order condition of the optimization in Equation 1 to get

$$\begin{aligned} F(b^*)(\hat{p}-1) + f(b^*)(\hat{p}-b^*) &= 0 \\ (1-F(a^*))(1-\hat{p}) - f(a^*)(a^*-\hat{p}) &= 0 \end{aligned}$$

If $\hat{p} \in (0,1)$ and $f(x) > 0$ for all $x \in (0,1)$, the existence and uniqueness of optimal $b^* < \hat{p} < a^*$ are guaranteed. When F and f are well-behaved smooth functions (as is the case for our experiments where F is a normal distribution), the optimal values can be solved quickly by numerical root-finding techniques.

4.3 Optimal log-utility policy

Following the procedure outlined in Section 3, we computed the optimal value and policy functions for several different parameterizations of wealths and beliefs for both Kelly and risk-neutral market makers.

We begin by considering the case where the market maker's private belief aligns with the beliefs of the traders. Figure 4 shows the optimal bid and ask prices over the series of traders when the market maker has wealth $(100, 100)$ (thickest line), $(50, 50)$ (medium line), and $(25, 25)$ (thinnest line). That is, the plot shows $\pi(t, (w, w))$ for $t \in \{1, \ldots, 50\}$ and $w \in (25, 50, 100)$.

Here, prices throughout the interaction are very close to the myopic optimization for the last trader, and the prices are very similar for all of the sampled wealths. In this scenario, the prices are also essentially equivalent to the optimal policy of a risk-neutral market maker.

In contrast, Figure 5 shows the optimal policies when the market maker's belief is $p = 0.6$ (shown by the cross-hatched line). This value is two standard deviations higher than the mean of the traders' beliefs. The policies are calculated at the same wealths as in Figure 4, that is, the policy of a market maker with wealth of 25, 50, and 100 in both states at every time step.

Unlike in the previous figure, the optimal policies change over time and are wealth-dependent. In this scenario, the optimal risk-neutral policy is a bid of 0.52 and an ask of 0.62. Because with large wealth a logarithmic utility market maker making small bets can be approximated well by linear utility, we know that as wealth increases, the market maker's optimal policy throughout the trading period will converge to be the optimal linear utility policy. How-

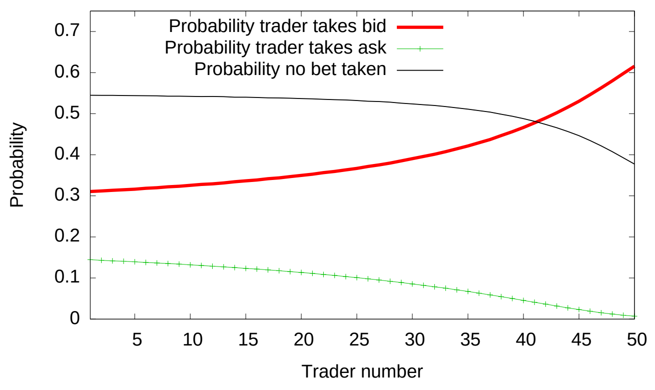


Figure 6: The probability of a trader taking each offered bet from a market maker with 25 wealth in each state over the entire trading period.

ever, at the smaller levels of wealth in our experiments, for all except the last few traders, the asking price for a unit contract is below the market maker's belief that the event will occur. Thus, for much of the trading period, from a myopic perspective the Kelly criterion market maker offers irrational bets.[1]

On the surface, this result seems confusing and even paradoxical. To see why it is the optimal policy, consider Figure 6, which displays the probability of each trader taking the bets offered by a market maker with a (constant) wealth of 25 in both states. It shows how the probability of a trader selling at the bid price rises over time, while the probability of a trader buying at the ask price falls. The first trader is about twice as likely to sell at the bid price than to buy at the ask price, while the last trader is about *87 times* more likely to sell than to buy.

For early traders, the market maker's bid and ask prices are roughly centered around 0.5, just like the distribution of agent beliefs. Con-

[1]One might think that this phenomenon could be explained by the market makers accumulating wealth from spread profits, and therefore becoming absolutely less risk-averse over time. However, even market makers with considerably larger endowments than could possibly be made through spread profits still display the same qualitative behavior.

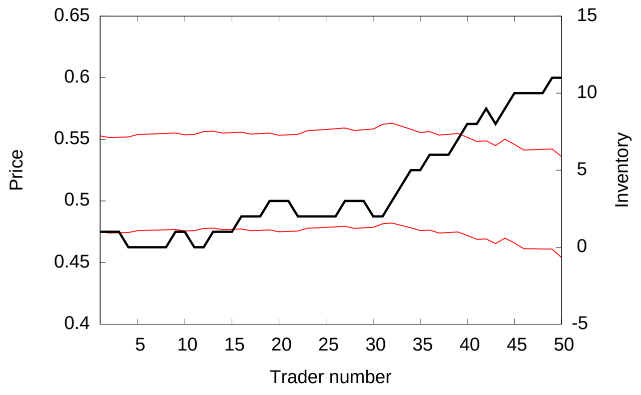


Figure 7: Simulating the prices (left axis; thin lines) and net inventory (right axis; thick black line) that result from the interaction of an optimal log**-utility market maker starting with 25 wealth in both states. In this figure, both the inventory and prices change over time.**

sequently, the market maker has a reasonable chance of matching traders' bids and asks and thus profiting off the bid/ask spread. A market maker that successfully matches the bids and asks of traders books a profit regardless of the personal beliefs of the market maker, even if those beliefs are, as in this scenario, very different from the prices in question. Of course, as fewer traders remain the setting more and more resembles a myopic optimization where, with equivalent wealths in both states, the market maker will employ a myopically rational strategy. This sophisticated policy emerges solely from the introduction of risk-aversion to the termination state, because a risk-neutral market maker in an identical setting displays none of this behavior.

Once the optimal policy is computed, we can simulate the behavior of the market maker against the pool of traders. Figure 7 shows the simulated prices of wealth 25 market maker in a sample interaction for the case where the the market maker has belief 0.6 and agents have belief mean of 0.5 (i.e., the setting for Figures 5 and 6). The thin lines, with values marked by the left axis, show the ask (upper line) and bid (lower line) prices of the market maker. The thick black line, with values market by the right axis, shows the market maker's net inventory, i.e., the market maker's payoff if the event occurs. The values on Figure 7 show the prices faced by trader i but the inventory *after* the participation of the trader. (The inventory line starts at 1 in this case because the first trader took the market maker's bid.) In this simulation, the market maker's expected utility from their wealth vector increases from 3.22 before any traders participate to 3.27 after all 50 traders participate.

Recall that in this setting the market maker has a significantly higher belief that the event will occur than does the pool of agents, so it is natural for the market maker to accumulate inventory. As the market maker accumulates inventory, its prices fall. This is because a risk-averse market maker prefers to take a small sure profit (the bid/ask spread from matching orders) over a somewhat larger speculative gain (from holding inventory). Consequently, the prices from the simulation are very different than the prices in Figure 5, because the prices in Figure 5 captured the prices of a market maker with constant wealth in both states over time. If the market maker were not taking on inventory, its prices would rise, as in Figure 5, but because the market maker in our simulation takes on inventory, that price rise is effectively dampened. Observe that in this simulation, because the price rise is dampened, the market maker's asking price is always less than 0.6 and therefore is myopically irrational for the entire trading period!

5. CONCLUSIONS

We initiated the study of rational market making where the market maker has knowledge about the probabilities of future events transpiring and of traders accepting bets that the market maker could offer them. We gave a general description of the optimization problem faced by a monopolistic market maker trying to maximize their expected utility. We investigated two cases in detail: a risk-neutral market maker (linear utility), that yields the highest expected wealth, and a Kelly criterion market maker (logarithmic utility) that yields the highest expected median and mode of wealth.

We showed that for a two-event setting, computing the optimal policy of a risk-neutral market maker is trivial, but computing the optimal policy of a log-utility market maker is not straightforward. Because there is no closed-form expression for the value function, we approximated it using Constantini shape-preserving interpolation. This interpolation technique preserves the concavity, monotonicity, and partial derivatives of the original function. Because it retains the shape of the approximated function, it is much more accurate than a simple grid interpolation. Since it is more accurate, to preserve the same level of accuracy we were able to solve the problem using a grid that is orders of magnitude coarser. Consequently, the time spent calculating the value function at each iteration of the dynamic program is orders of magnitude faster using the more-sophisticated shape-preserving interpolation technique than with a simple linear interpolation. Because our problem is defined over the whole positive orthant while the shape-preserving technique we used works only over a finite grid, we had to develop an extension of the technique at the lower and upper boundaries of the grid.

We showed that the optimal policy for a risk-neutral market maker is always myopically rational. In contrast, our experiments showed that the optimal policy for a Kelly criterion market maker is often myopically irrational, and that a Kelly criterion market maker could have a myopically irrational policy for the entire trading period. We showed evidence that a log-utility market maker would begin the trading period pricing in order to capture the bid/ask spread from agents. Recall that the traders' response to the market maker's prices is a random variable, so that whether or not a market maker acquires inventory is stochastic, not deterministic. If that pricing resulted in the market maker not taking on inventory, our results showed the market maker would gradually transition from pricing to capture the bid/ask spread to myopically pricing based on their beliefs. However, if the policy in early periods resulted in the market maker taking on inventory, we showed that the optimal pricing throughout the interaction need not deviate much from initial prices. Therefore, depending on how traders react to the market maker's policy, a Kelly criterion market maker could follow a myopically irrational policy for the entire trading period.

In sum, we can distill our results into three qualitative suggestions for monopolistic Kelly criterion market makers with good prior information facing a stream of anonymous traders, as in Internet sports betting: (1) change odds as bets are received and wealth changes, (2) begin pricing with the goal of matching orders and procuring a bid/ask spread, and (3) gradually transition into pricing to accumulate the market maker's desired inventory position.

There are several extensions to consider to our framework. Our model assumed the market maker was monopolistic, so that it could maximize profit without fear of competition. One extension could be to examine a competitive setting between several risk-averse market makers.

Another extension would be to incorporate informed traders into the pool of trading agents. These agents could have correct knowledge about the future, but, more importantly, the market maker could know of and react to their existence. The presence of informed traders that influence the market maker in this way would make our setting much closer to the Bayesian market maker setting explored in Das and Magdon-Ismail [2009], but would be even more complex because of the risk aversion of the market maker.

While our framework applies to any number of events and bets, our computational experiments focused on the binary case. An extension would be to develop algorithms to solve for optimal policy with multiple events. The Constantini shape-preserving technique used in this paper could presumably be applied to more than two events, although it will suffer from the curse of dimensionality. Perhaps an alternate approach to approximating the value function could be used in this case, such as a spline of radial basis functions, although this would be unlikely to preserve the monotonicity and concavity of the value function and so could lead to poor or unrealistic policies.

Acknowledgements

This work was supported by NSF grants CCF-1101668, IIS-0964579, and IIS-0905390, and by the Google PhD Fellowship in Market Algorithms.

References

S. Agrawal, E. Delage, M. Peters, Z. Wang, and Y. Ye. A unified framework for dynamic pari-mutuel information market design. In *ACM Conference on Electronic Commerce (EC)*, pages 255–264, 2009.

A. Ben-Tal and M. Teboulle. An old-new concept of convex risk measures: The optimized certainty equivalent. *Mathematical Finance*, 17(3):449–476, 2007.

M. Chakraborty, S. Das, A. Lavoie, M. Magdon-Ismail, and Y. Naamad. Instructor rating markets. In *Conference on Auctions, Market Mechanisms and Their Applications (AMMA)*, 2011.

Y. Chen and D. M. Pennock. A utility framework for bounded-loss market makers. In *Proceedings of the 23rd Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 49–56, 2007.

Y. Chen and D. M. Pennock. Designing markets for prediction. *AI Magazine*, 31(4), 2010.

Y. Chen, D. Reeves, D. Pennock, R. Hanson, L. Fortnow, and R. Gonen. Bluffing and Strategic Reticence in Prediction Markets. In *WINE*, 2007.

Y. Chen, S. Dimitrov, R. Sami, D. M. Reeves, D. M. Pennock, R. D. Hanson, L. Fortnow, and R. Gonen. Gaming prediction markets: Equilibrium strategies with a market maker. *Algorithmica*, 58: 930–969, December 2010.

P. Constantini and F. Fontanella. Shape-preserving bivariate interpolation. *SIAM J. Numer. Anal.*, 27:488–506, April 1990. ISSN 0036-1429.

S. Das. The effects of market-making on price dynamics. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 887–894, 2008.

S. Das and M. Magdon-Ismail. Adapting to a market shock: Optimal sequential market-making. In *Advances in Neural Information Processing Systems 21 (NIPS)*, pages 361–368, 2009.

D. De Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, pages 850–865, 2003.

S. Dimitrov and R. Sami. Non-myopic strategies in prediction markets. In *EC '08: Proceedings of the 9th ACM conference on Electronic commerce*, pages 200–209, 2008.

J. Feigenbaum, L. Fortnow, D. Pennock, and R. Sami. Computation in a distributed information market. In *ACM Conference on Electronic Commerce (EC)*, pages 156–165, 2003.

L. R. Glosten and P. R. Milgrom. Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. *Journal of financial economics*, 14(1):71–100, 1985.

D. Gode and S. Sunder. Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. *Journal of Political Economy*, pages 119–137, 1993.

G. J. Gordon. Stable function approximation in dynamic programming. In *International Conference on Machine Learning (ICML)*, pages 261–268, 1995.

C. Guestrin, D. Koller, and R. Parr. Max-norm projections for factored MDPs. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 673–682, 2001.

K. Judd. *Numerical methods in economics*. The MIT Press, 1998.

J. Kelly Jr. A new interpretation of information rate. *IRE Transactions on Information Theory*, 2(3):185–189, 1956.

A. S. Kyle. Continuous Auctions and Insider Trading. *Econometrica*, 53(6):1315–1335, 1985.

A. Mas-Colell, M. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, New York, NY, 1995.

A. Othman. Zero-intelligence agents in prediction markets. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 879–886, 2008.

A. Othman and T. Sandholm. When Do Markets with Simple Agents Fail? In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 865–872, Toronto, Canada, 2010.

W. Poundstone. *Fortune's Formula: The Untold Story of the Scientific Betting System That Beat the Casinos and Wall Street*. Hill and Wang, 2006.

K. Rock. The specialist's order book and price anomalies. *Review of Financial Studies*, 1996.

J. Stachurski. Continuous state dynamic programming via nonexpansive approximation. *Computational Economics*, 31(2):141–160, 2008.

N. Stokey, R. Lucas, and E. Prescott. *Recursive methods in economic dynamics*. Harvard University Press, 1989.

S.-P. Wang and K. L. Judd. Solving a savings allocation problem by numerical dynamic programming with shape-preserving interpolation. *Comput. Oper. Res.*, 27:399–408, April 2000. ISSN 0305-0548.

Can a Zero-Intelligence Plus Model Explain the Stylized Facts of Financial Time Series Data?

Imon Palit
SEFeMEQ Department
University of Rome
Tor Vergata
palit@uniroma2.it

Steve Phelps
Centre for Computational
Finance and Economic Agents
(CCFEA)
sphelps@essex.ac.uk

Wing Lon Ng
Centre for Computational
Finance and Economic Agents
(CCFEA)
wlng@essex.ac.uk

ABSTRACT

Many agent-based models of financial markets have been able to reproduce certain stylized facts that are observed in actual empirical time series data by using “zero-intelligence” agents whose behaviour is largely random in order to ascertain whether certain phenomena arise from market microstructure as opposed to strategic behaviour. Although these models have been highly successful, it is not surprising that they are unable to explain *every* stylized fact, and indeed it seems plausible that although some phenomena arise purely from market micro-structure, other phenomena arise from the behaviour of the participating agents, as suggested by more complex agent-based models which use agents endowed with various forms of strategic behaviour. Given that both zero-intelligence and strategic models are each able to explain various phenomena, an interesting question is whether there are hybrid, “zero-intelligence plus” models containing a minimal amount of strategic behaviour that are simultaneously able to explain all of the stylized facts. We conjecture that as we gradually increase the level of strategic behaviour in a zero-intelligence model of a financial market we will obtain an increasingly good fit with the stylized facts of empirical financial time-series data. We test this hypothesis by systematically evaluating several different experimental treatments in which we incrementally add minimalist levels of strategic behaviour to our model, and test the resulting time series of price returns for the following statistical features: fat tails, volatility clustering, persistence and non-Gaussianity. Surprisingly, the resulting “zero-intelligence plus” models do *not* introduce more realism to the time series, thus supporting other research which conjectures that some phenomena in the financial markets are indeed the result of more sophisticated learning, interaction and adaptation.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

1. INTRODUCTION

Many agent-based models of financial markets have made use of zero-intelligence agents in order to explain various phenomena in microeconomics and finance. In these models, market scenarios are simulated with agents whose behaviour is largely random. They are particularly useful in attempting to attribute whether a particular phenomenon is caused by the market mechanism or to the strategic behaviour of participating traders, since any non-random regularities that appear in the macroscopic behaviour of such a model can be attributed to the mechanics of the underlying market microstructure rather than to the behaviour of the agents.

Zero-intelligence models have been applied to studying phenomena in both economics and finance. The concept was originally introduced in the field of microeconomics in order to explain how traders were able to converge on equilibrium prices in a continuous double-auction trading environment: Gode and Sunder set out to determine whether this phenomena could be ascribed to the intelligence of the human traders, or alternatively whether the efficiency of the market was due to the trading institution itself [7]. They did so by introducing one of the first agent-based simulation models in which agents with various levels of *random* price setting behaviour were simulated under the same market microstructure rules as used in the original experiment with human traders. They found that they were able to reproduce convergence to equilibrium prices (under certain treatments), suggesting that the efficiency of the market was due to the “intelligence” of the market institution itself, rather than intelligent behaviour on the part of the participants.

However, in later work Cliff and Bruten [4] showed that under different treatments Gode and Sunder's zero-intelligence agents were not able reproduce the original convergence results (“zero is not enough”), and that so called zero-intelligence *plus* (ZIP) agents with a minimal level of *conditional* bidding behaviour were required in order to converge to equilibrium prices under a wider variety of initial conditions.

Although zero-intelligence models originated in the field of microeconomics, there have been considerable successes in using them to explain the stylized facts of *financial* time series data. For example, one such stylized fact is positive correlation in order flow: the probability of observing a given type of order in the future is positively correlated with its empirical frequency in the past. This phenomena was originally documented by Bias et al. on the Paris Bourse exchange [1]. [8] was able to reproduce this stylized fact using a zero-intelligence agent-based model, thus suggesting that empirical regularities in order-flow arise from the operation of the market mechanism rather than the behaviour

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

of traders.

However, it would be naive to suppose that *all* phenomena observed in financial time series data could be reproduced using models in which traders behave entirely randomly, and indeed it seems apriori reasonable that although some phenomena arise purely from market micro-structure, other phenomena arise from the behaviour of the participating agents, as suggested by more complex agent-based models which use agents endowed with various forms of strategic behaviour [3].

An interesting question then arises as to what is the minimal level of intelligence required for an agent-based model to reproduce statistically-realistic time series data. Analogous to Cliff's "zero is not enough" conjecture, we hypothesise that by gradually increasing the level of strategic behaviour in a zero-intelligence model we will obtain an increasingly good fit with the empirical data of financial markets. We test this hypothesis by systematically evaluating several different experimental treatments in which we incrementally add minimal amounts of conditional behaviour to our model, and test the resulting time series of returns for the following stylized facts: fat tails, volatility clustering, high-frequency persistence and non-Gaussianity [5][12].

Stylized facts are statistical regularities that are often found in real market data across different markets and different periods of time [5]. By reproducing these empirically observed stylized facts we can validate our model and by building a microscopic (trader-based) model of the market attempt to understand what assumptions are consistent with these properties. We test whether minimal strategic behaviour is sufficient to reproduce non-Gaussian distributions (characterised by excess kurtosis), fat tailed distributions (characterised by power law decay in the tail of the distribution), high frequency persistence (characterised by super-diffusive behaviour at short time scales) and volatility clustering (characterised by nonstationarity in price changes).

We start with a simple zero-intelligence model in which different *types* of order, as classified by the original Paris Bourse study [1] are submitted to the exchange. In our first experimental treatment order types are chosen from a discrete uniform distribution. We then gradually introduce additional complexity into the model by allowing the probability with which an event type is chosen to change in response to the state of the market – that is, we introduce *conditional* or strategic behaviour into our model. We do so systematically and in line with empirically-observed phenomena in real financial markets.

The paper is structured as follows. In sections 2 and 3 we give an overview of empirically documented conditional order flow phenomena in financial time series data. In section 4 we give detailed description of our model, including the event types and detailed descriptions of the seven experimental treatments used in our investigation. In section 5 we describe our methodology. In sections 6 and 7 we show and discuss our results, and finally we conclude in section 8.

2. ABSOLUTE AND CONDITIONAL ORDER FLOW

[1] show that there is a "diagonal effect" in high-frequency financial time-series data: viz., order flow is conditional on past order flow. Both [1] and [2] show there is a "stimulated refill" liquidity process: order flow is conditional on the state of the book. Previous analysis by [13] equating effective costs of market and limit orders, demonstrated that a necessary condition for statistical arbitrage efficiency is a linear relationship between impact, bid-ask spread and volatility. [13] empirically found this relationship to hold true and propose this as a market law or stylized fact that has a theoretical underpinning based on ecology between liquidity takers and providers in pure order driven markets. They argue this phenomenon emerges as liquidity in an order-driven market self-organises toward statistical efficiency. Interestingly this theoretical motivation occurs endogenously without any role of information and does not espouse the traditional view of efficient market theory: that of informationally-efficient markets.

Using our model we investigate alternative treatments corresponding to both conditional and unconditional order flow. In each case, we test for fat tails, volatility clustering, long-memory and non-Gaussianity in returns. We do this by simulating the effects of adding the "diagonal effect" and "stimulated refill" liquidity process to a zero-intelligence model, thus gradually introducing strategic behaviour into the model.

Using nineteen days of data from the Paris Bourse in November 1991, [1] calculated the frequencies of events classified depending on aggressiveness[1], and our model builds on this classification. On the buy-side orders can be categorised into seven event types corresponding to differing levels of aggressiveness as detailed below (and summarised in Table 1).

A `Large Buy` event represents market orders that take liquidity through the best ask and through more than one price level by walking up the book. A `Market Buy` removes all orders at the best ask exactly. A `Small Buy` represents a market order to buy a quantity lower than that offered at the best ask. `New Bid Within` events are limit orders to buy posted within the best bid and ask quotes inside the spread. A `New Bid At` event posts a new bid limit order at best quotes joining the queue at the best bid and a `New Bid Away` posts a bid limit order at a price below the best bid. Finally a `Cancel Bid` refers to cancellations of bid limit orders irrespective of price level.

An additional seven event types can be defined analogously on the sell side of the book (storing the asks). Including off-book trades, we therefore have a total of fifteen different types of event. The results from the Paris Bourse study are summarised in Table 1 which shows the empirical probability distribution over the event types, with the probabilities adjusted to exclude applications[2]. One important aspect to note is that their analysis is based on data that only gave the five best bid and ask price step levels of the book.

Bias et al. went on to analyse the same empirical probabilities over event types *conditional* on the previous event. The rescaled conditional probabilities from [1] are given in Table 2. Again we rescale their original results to exclude applications[3].

[1]By aggressiveness we mean the immediacy to which the agent is willing to trade.

[2]We do this as we are not interested in off-book trades. The probabilities are rescaled to sum up to 100%.

[3]The readjusting is again a simple rescaling to exclude applications so that the column probability vectors sum up to 100%. It does however ignore events at $t + 2$ if an application occurred at $t + 1$ e.g. the

Conditional frequencies of limit order book events

t-1	Large Buy	Market Buy	Small Buy	New Bid Within	New Bid At	New Bid Away	Cancel Bid	Large Sell	Market Sell	Small Sell	New Ask Within	New Ask At	New Ask Away	Cancel Ask
Large Buy	**7.51**	**3.51**	15.80	**13.45**	5.16	6.08	5.37	1.85	1.14	14.83	7.48	4.79	7.04	**5.99**
Market Buy	3.26	2.52	**17.45**	2.18	7.02	**11.55**	**8.08**	3.42	**6.28**	23.95	2.83	2.50	4.94	4.02
Small Buy	**3.80**	**2.96**	**21.00**	9.91	5.40	6.49	4.12	2.41	1.79	20.26	6.03	4.65	6.35	4.84
New Bid Within	**3.27**	2.19	13.43	**13.94**	**7.03**	**9.43**	5.34	3.08	2.03	19.02	8.48	2.99	5.55	4.21
New Bid At	2.27	2.45	14.29	**16.85**	**7.02**	7.50	5.03	1.64	0.92	23.09	6.17	3.92	5.40	3.47
New Bid Away	1.94	1.63	11.72	7.65	**8.14**	**18.59**	4.54	3.29	2.46	20.38	7.05	3.44	5.69	3.49
Cancel Bid	2.33	1.76	10.85	12.79	5.35	8.27	**9.97**	3.90	2.32	20.08	8.94	3.67	5.95	3.82
Large Sell	0.94	0.95	6.62	9.05	4.66	6.70	**6.12**	**10.09**	**3.42**	**24.52**	12.89	3.42	5.12	5.51
Market Sell	2.93	**6.41**	**16.37**	2.31	2.96	4.34	3.65	**4.88**	**3.08**	**29.73**	2.24	5.04	**9.62**	**6.45**
Small Sell	1.62	1.63	10.49	7.07	4.88	5.93	4.47	**4.93**	3.02	**34.61**	8.05	4.09	5.41	3.79
New Ask Within	2.05	1.91	10.08	9.64	4.12	6.17	4.15	4.54	2.66	23.17	**12.89**	**5.65**	**7.93**	5.05
New Ask At	1.44	0.99	13.04	6.54	4.91	5.76	3.91	3.06	2.53	23.56	**15.03**	**6.78**	7.44	5.02
New Ask Away	2.29	2.2	11.04	7.39	4.21	5.98	3.43	2.87	2.14	21.80	5.97	**7.22**	**19.77**	3.68
Cancel Ask	2.08	1.90	10.20	8.95	4.50	6.25	4.62	4.16	2.18	19.67	**14.19**	4.05	7.76	**9.47**

Table 2: Conditional order and trade event frequencies rescaled from empirical results in [1, p. 1673]. Each entry in the main body represents a probability of an order book event (columns) conditioned by the previous event (rows). The three highest values in a given column are emphasised to illustrate the "diagonal effect".

Book Side	Event Type	Probability(%)
	`Large Buy`	2.4
	`Market Buy`	2.1
	`Small Buy`	12.7
Buy Side	`New Bid Within`	9.2
	`New Bid At`	5.4
	`New Bid Away`	7.6
	`Cancel Bid`	4.8
	`Large Sell`	3.9
	`Market Sell`	2.4
	`Small Sell`	24.5
Sell Side	`New Ask Within`	8.5
	`New Ask At`	4.4
	`New Ask Away`	7.1
	`Cancel Ask`	4.9

Table 1: Absolute Frequencies of Limit Order Book Events rescaled from empirical results from the Paris Bourse [1, p. 1670].

As was noted in [1] the numbers on the diagonal tend to be larger than others in the same column. This "diagonal effect" suggests an event type is more likely to be preceded by the same event type than any other. Table 2 illustrates this along with the three most likely preceding events for each event highlighted in bold to show this diagonal effect.

3. STIMULATED REFILL

Along with the "diagonal effect" another interesting feature of order flow is the "stimulated refill" process ([2]) in order-driven markets where liquidity amongst liquidity providers and takers is self-organised to eliminate statistical arbitrage. In [1] they found most limit order activity being at or within quotes despite there being least depth at the best quotes when looking at the average profile of the order-book over time. They showed this to be a result of a cycle of transient depth whereby when spreads were tight, trades were more frequent thus widening the spread, followed by more within and at-quote limit order activity when spreads and depth at best bid or offer (BBO) were large. [1] and [2] argue that this behaviour is due to the liquidity replenishing and under-cutting/out-bidding activities of market-makers.

Table 3 below shows a stylised example set of probability vectors that show this effect of dynamic liquidity switching between regimes of submitting more trades and less orders versus a regime of submitting more orders and less trades dependant on the bid-ask spread. The probabilities are based on the empirical probabilities that [1] found when classifying the state of the book in a high or low spread regime in relation to the median value for the bid-ask spread.

In order to use these results in the context of a simulation model, several modifications are required. First all probability vectors are again rescaled to exclude applications which are non-order book events. Secondly when doing this particular analysis [1] grouped all their previous trade events classifications as just one trade event (e.g. all `Large Buy, Market Buy, Small Buy` events become classified just as a buy trade event). So in Table 3 we split the probabilities back into the more granular trade classifications by assuming they have the same relational proportions (amongst the three trade classifications on the relevant side of the book) as in the absolute case from Table 1. This maintains a probability for all fourteen events that will be needed for the model we develop in the next section. The last transformation averages their results that were based on further classifying the book as in a high or low depth regime as well as a high or low spread regime. This results in reducing their four probability vectors to two based simply on a high or low spread. As the fourteen order book event classifications from [1] only affect traded prices by shifting price levels of the order book, the volume depth at particular levels is not meaningful in the context of the model we will develop; therefore the issue of volumes of order book events is simplified in our analysis for now.

Thus in Tables 1, 2 and 3 we have three types of dynamics of order books that can be investigated by simulation of events based on the probability vectors.

4. THE MODEL

Our model is based on the event classification of the Bias et al. study [1] described in Section 2. Trading occurs over a series of discrete simulation steps. During each step an event type is chosen from Table 3, and we then simulate its effect on the order-book. We note any resulting trades and record the time series of transaction prices for later analysis. The

event chain: `Large Market Buy (t)` $\rightarrow$`Application (t+1)` $\rightarrow$`Small Sell (t+2)` is not rescaled as `Large Market Buy (t)` $\rightarrow$`Small Sell (t+1)`. Similarly it ignores any event sequence chains with a block of intermediary applications e.g. `Event (t)` $\rightarrow$`Application (t+1) ...`$\rightarrow$`Application (T-1)` $\rightarrow$`Event(T)` etc. However we find this makes negligible differences to the probabilities.

Event Type	Small Spread Probability (%)	Large Spread Probability (%)
`Large Buy`	3.0	2.0
`Market Buy`	2.6	1.7
`Small Buy`	15.7	10.3
`New Bid Within`	5.1	13.9
`New Bid At`	4.4	6.2
`New Bid Away`	7.0	8.5
`Cancel Bid`	4.9	5.0
`Large Sell`	4.5	2.9
`Market Sell`	2.8	1.8
`Small Sell`	28.5	18.1
`New Ask Within`	4.6	13.0
`New Ask At`	4.3	4.8
`New Ask Away`	7.2	7.4
`Cancel Ask`	4.7	4.5

Table 3: Dynamic frequencies of limit order book events (rescaled and modified from empirical results from [1, p. 1677] on the Paris Bourse in 1991).

event types are classified according to their liquidity-taking or liquidity-providing nature.

Our goal is to simulate this model under different experimental treatments, starting with a zero-intelligence treatment in which the type of event is chosen unconditionally at random from a discrete uniform distribution, followed by treatments which simulate more complex behaviour by allowing the probability with which a particular event is generated to be conditional on the current state of the market. This enables us to test the affect of conditional order flow adapting to either previous order flow or the state of the book; that is, to analyse the effect of *strategic* behaviour. ha Similarly to other models in the literature ([3, 9]), we use a volume size of 1 for all events except for the following: `Large Buy, Large Sell, Market Buy, Market Sell`. This is because these events are defined by removing a price step level of the order book at the best opposite quote and therefore need to have a volume corresponding to the available liquidity at best quotes. Prices of orders are rounded to a tick size of Δ^{Tick} and if the book is empty (e.g. at the start) and there is no best bid and offer (BBO) limit orders are posted with proximity to a reference fundamental price of P_f.

4.1 Market Events

In this section we detail how the simulated events introduced in Section 2 specifically affect the limit order book in our simulation model. We call b_t the best bid and a_t the best ask in the book at the current time of the simulation. All the event types can be classified as either market orders (`Large Buy`, `Large Sell`, `Market Buy`, `Market Sell`, `Small Buy`, `Small Sell`) or as limit orders (`New Bid Within`, `New Ask Within`, `New Bid At`, `New Ask At`, `New Bid Away`, `New Ask Away`, `Cancel Bid`, `Cancel Ask`)[4]. We now go through both of these two main order types detailing the model's market mechanics for all event types when simulated in our model:

4.1.1 Market Order Events

Whenever a `Market Buy` event occurs, the volume at the best ask price is cleared out: all sell orders currently queued on the order-book at this price step are removed (and vice-versa for `Market Sell` events at the best bid). This could still be a volume of 1 (similar to `Small Buy` and `Small Sell` events) if there is not a queue at the BBO. Trade events (`Large Buy`, `Market Buy`, `Small Buy`, `Large Sell`, `Market Sell` and `Small Sell`) occur if and only if there is available liquidity on the relevant book side. If not, the book remains unchanged and the simulation moves on to the next time tick.

For a `Large Buy`, a market order is submitted with volume equal to the total volume at the best ask +1 so that it removes liquidity at the best ask (a_t) and then trades at the next price level (a_t^2) above the best ask. A `Large Sell` event can be defined analogously taking liquidity on the bid-side of the book.

4.1.2 Limit Order Events

For limit orders submitted within the spread we assume they are posited randomly in between the spread i.e. uniformly deposited in the interval [b_t, a_t]. Limit orders posted at best quotes enter orders with a size of 1 at the end of the queue at BBO (b_t for `New Bid At` events, a_t for `New Ask At` events). Limit orders posted away from the spread are submitted uniformly anywhere between [b_t^5, b_t] for `New Bid Away` events and [a_t, a_t^5] for `New Ask Away` events where b_t^5 and a_t^5 represent the 5th best bid and 5th best ask level in the book. We do this as the probabilities are based on the empirical results of [1] who used order book data up to five price levels. If in the simulation there aren't at least five price levels of liquidity in the book, the limit orders are posted uniformly in [b_t-D, b_t] for bids and [a_t, a_t+D] for asks where D represents a maximum offset parameter in the model. Limit order cancellations (`Cancel Bid, Cancel Ask`) remove at random an existing limit order in the book on the relevant side of the book. All limit orders on the relevant book side have equal probability of being deleted.

4.2 Experimental Treatments

We simulate our model under seven different experimental treatments. Each treatment is described in detail in the sections below and summarised in Table 4. The purpose of simulating the model under these different experimental treatments is to systematically evaluate whether incrementally introducing conditional behaviour into a zero-intelligence model produces more realistic time series data.

4.2.1 Equal treatment ("equal")

Under this treatment, the event types are chosen randomly from a discrete uniform distribution over all fourteen event types. That is, the probability of generating any given type of event is $\frac{1}{14}$. This is the simplest treatment and corresponds to a pure "zero-intelligence" model.

4.2.2 Absolute treatment ("abs")

Under this treatment the underlying distribution used to generate events remains static, but is calibrated with probabilities based on empirical data: that is, the probability of each event is given by the unconditional frequencies in the empirical analysis of Bias et al. on the Paris Bourse [1]. Thus the order book events are simulated with an unconditional discrete empirical probability distribution using the probability vector from Table 1. This corresponds to a calibrated "zero-intelligence" (ZI) treatment.

[4]There are no marketable limit orders as limit order event types aren't defined this way. In the original [1] analysis marketable limit orders would be counted as market orders.

Section	Treatment Name	Experiment Conditions
4.2.1	*"equal"*	Equal Event Probabilities
4.2.2	*"abs"*	Absolute Event Probabilities
4.2.3	*"cond"*	Conditional Event Probabilities
4.2.4	*"symmetric"*	Symmetric Unconditional Event Probabilities
4.2.5	*"dynamic"*	Regime Probabilities based on fixed spread
4.2.6	*"dynamicAlt"*	Regime Probabilities based on localised window
4.2.7	*"cluster"*	Regime Probabilities alternating after random horizons

Table 4: Summary of experimental treatments

4.2.3 *Conditional treatment ("cond")*

Under this treatment events are generated from a *conditional* discrete empirical probability distribution in which probabilities are conditioned on the previous event type. The probability vectors used are those in Table 2 and allows switching between 14 different probability distributions. For example if the last event simulated in the book was a `New Bid Within` the probability vector used to simulate the next event would be the 4th row in Table 2. The conditional order book event frequencies are rescaled from empirical results in [1, p. 1673]. This enables us to test the effects of adding the "diagonal effect" from [1]. Since behaviour in this model is conditional on previous order flow, we can ascribe to it a minimal level of strategic behaviour.

4.2.4 *Symmetric treatment ("symmetric")*

Under this treatment events are generated from a static probability distribution, but with symmetric probabilities by averaging the unconditional empirical probability distribution to be symmetric on both buy and sell sides of the book. For example from Table 1 we can see in the "abs" case the probability for a `Large Buy` is 2.4% and for a `Large Sell` is 3.9%. In the "symmetric" case the probability of a `Large Buy` would equal the probability of a `Large Sell` with the value being $\frac{2.4\%+3.9\%}{2} = 3.15\%$. This is done with all 7 pairs of events that occur on the buy and sell side of the book.

4.2.5 *Dynamic treatment ("dynamic")*

This treatment has probabilities conditioned on the state of the book choosing from the two spread regimes in Table 3. These dynamic frequencies are rescaled and modified from empirical results from [1, p. 1677]. It allows us to test adding the "stimulated refill" liquidity process to the unconditional ZI framework of the "abs" treatment whereby more liquidity taking occurs when spreads are low and more liquidity providing activity when spreads become high. We run this treatment after the "abs" treatment and use the average median spread value from that case as a fixed endogenous comparison value to switch between the high and low spread probability vectors depending if the spread at the current time tick is more or less than this comparison spread value respectively.

4.2.6 *Alternative Dynamic treatment ("dynamicALT")*

This treatment is a variant of the "dynamic" treatment in which the comparison spread value alters and is calculated locally (from within the "dynamicALT" simulation run) as the median value from a localised rolling window of spreads in the last 100 simulation steps. Again, the probability distributions are given by Table 3.

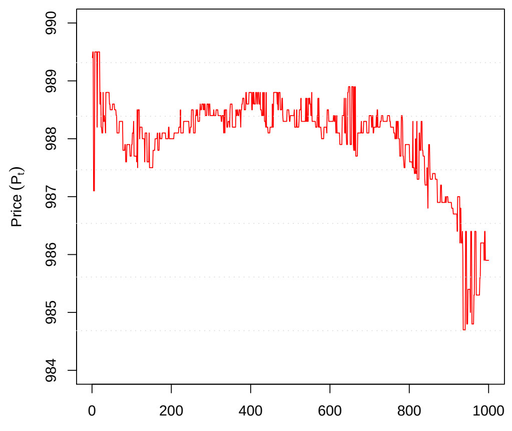


Figure 1: A sample price path for 1,000 time steps for the "abs" treatment

4.2.7 *Cluster treatment ("cluster")*

In this treatment the dynamic switching between the two liquidity regimes is introduced exogenously in the simulation as the model simulates alternating between the two probability regimes (low to high to low etc..) from Table 3 with each regime lasting a L number of simulation steps where L is initialised at the start of each new regime by drawing a uniform random number between 1 and 100.

5. METHODOLOGY

For each of the above treatments we simulate the outcome on the order book of the events generated by the model.

At the start of the simulation there is no best bid or offer (BBO), therefore the initial market price at time $t = 0$ was set to the fundamental price $P_f = 10^3$. The tick size Δ^{Tick} was set to 10^{-1} and the maximum offset parameter $D = 5$ (which corresponds to 50 ticks). Each simulation was run for 1.1×10^4 time steps. We recorded prices only from $t > 10^3$ onwards in order to wash out any initial effects.

We used the Mersenne Twister ([10]) algorithm to draw all random variates in the simulation. We ran a total of seven experimental treatments which are summarised in Table 4. For each treatment we ran 10^4 independent simulations, thus producing a total of 7×10^4 different time series of prices and returns, which were then analysed for the stylized facts.

Figure 1 shows a typical section of the time series of prices

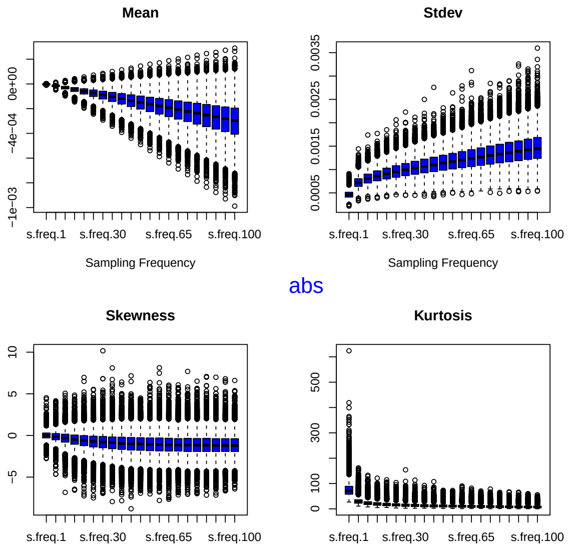


Figure 2: Box-Whisker plot for the Mean, Standard Deviation, Skewness and Kurtosis of returns from the "abs" treatment.

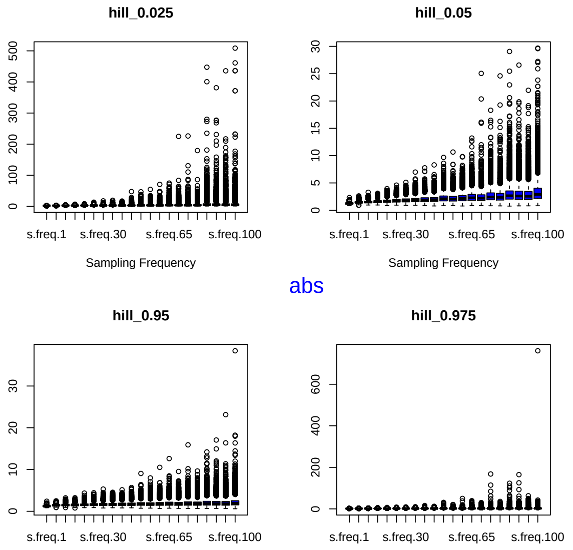


Figure 3: Box-Whisker plot for the Hill Estimator of returns from the "abs" treatment at the lower $(0.025, 0.05)$ and upper $(0.95, 0.975)$ quantiles.

produced over the course of 10^3 simulation steps. From the price time series we calculated the time series of returns r_t according to the following equation:

$$r_{t_1} = log(p_{t_1}) - log(p_{t_0}), \tag{1}$$

where p_t is the market price (defined as the last trade price) at time t as measured in discrete "event time": that is, the time value is incremented by 1 each time an order book event occurs and represents the number of simulation steps that have elapsed since the start of the simulation. Prices are sampled at regular time intervals $t_1 - t_0$.

We then tested for the presence of the following statistical features in the simulated time series of returns as given by equation 1: fat tails, volatility clustering, persistence and non-Gaussianity. For each treatment we computed the (a) Hill-estimator, (b) Hurst exponent, (c) p-values for the ARCH-LM test and (d) p-values for the Jarque-Bera test. For the Hill-estimator the 0.025, 0.05, 0.95 and 0.975 quantiles were chosen. Because the return calculation depends on the frequency with which prices are sampled (t_1 - t_0), the tests were calculated for each sampling frequency in the set $\{5, 10, \ldots, 95\}$ as measured in simulation steps. A distribution of stylised fact test results is obtained from the 10^4 simulation runs which allows us to run t-tests comparing treatments.

6. RESULTS

We first show some basic statistical results from the returns on the "abs" treatment in Figure 2. We can see the mean returns are negative, as one might expect given the asymmetry in the empirical probability distribution: as can be seen from Table 1 sell side of the book events have higher probabilities than the corresponding buy side of the book events reflecting the empirical data collected in [1]. Again, as one might expect, as the sampling interval increases the returns exhibit higher standard deviations. We observe slight negative skewness and significant kurtosis suggesting that the "abs" treatment reproduces non-Gausianity and fat tails.

This is confirmed when looking at the Hill estimator in Figure 3 which tends to around 3 in the upper and lower tail. In Figure 4 we can see we get a Hurst exponent ≈ 0.6 which [9] and [12] argue to be realistic on short to medium timescales. From Figure 4 we can see the ARCH-LM test shows no evidence of volatility clustering (p-values much higher than a 5% significance level) but the Jarque-Bera test shows significant deviance from Gaussian returns (p-value close to 0).

We now turn to the results of performing the same statistical tests on the other treatments. Figure 5 compares the sample means of summary statistics run on the returns of the seven treatments described in Section 4.2 and summarised in Table 4. We can see in Figure 5 that apart from the "equal" and "symmetric" treatments all other treatments have a negative return due to asymmetry with higher probabilities for events on the sell side of the order book.

With the exception of the "equal" treatment all the treatments show significant kurtosis and reproduce non-Gausianity in returns. Figure 6 shows the sample means of the Hill estimator tests on the 7 treatments and again shows with the exception of the "equal treatments" all other treatments show similar behaviour reproducing fat tails in the returns. Figure 7 compares the Hurst Exponent, ARCH-LM test p-values and Jarque-Bera test p-values of returns for the 7 treatments and show all models reproduce realistic high frequency Hurst exponents ranging from 0.4 to 0.6. The ARCH-LM test shows like the "abs" treatment none of the treatments reproduce volatility clustering in returns. The Jarque-Bera test p-values confirm that with the exception of the "equal" treatment the other treatments exhibit non-Gaussian returns.

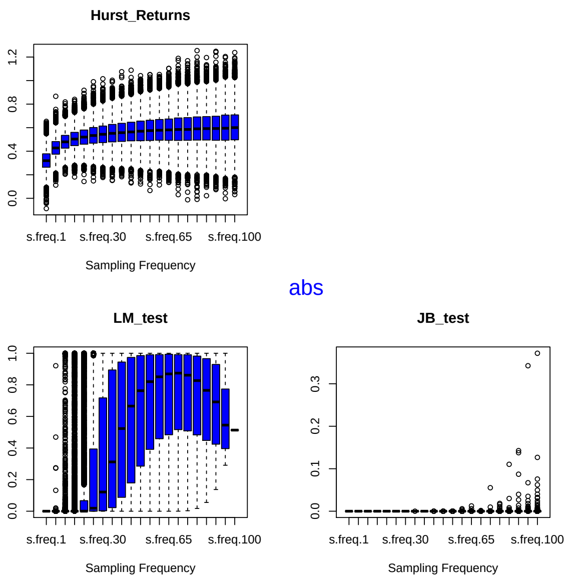


Figure 4: Box-Whisker plot for the Hurst Exponent, ARCH-LM test p-values and Jacques-Bera test p-values of returns from the "abs" treatment.

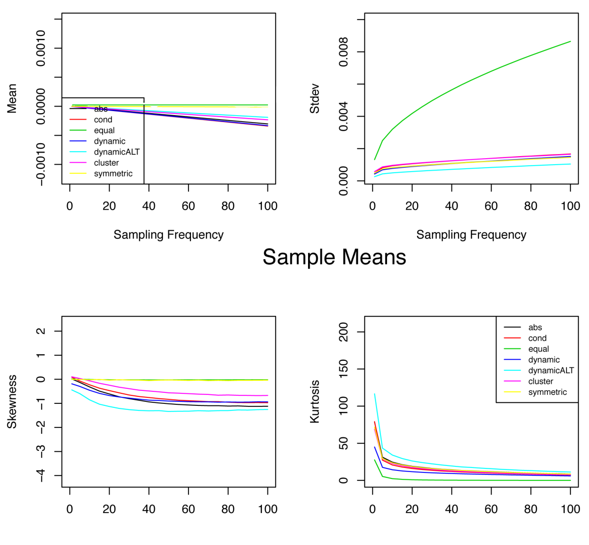


Figure 5: Sample means for mean, standard deviation, skewness and kurtosis of returns for the seven treatments.

7. DISCUSSION

In all experiments other than the "equal" treatment, we observe realistic Hurst exponents, fat tails and non-Gaussianity in return behaviour. Thus our results show that *unconditional* zero-intelligence order flow is sufficient to reproduce these phenomena, provided that the probability distribution used to drive the zero-intelligence behaviour is calibrated against empirical frequencies (as per the "abs" treatment).

In the treatments "cond", "dynamic", "dynamicAlt" and "cluster" we have simulated *conditional* order flow (a proxy for strategic behaviour) and found similar results to the unconditional order flow treatments. None of our experiments reproduced volatility clustering. This suggests that a more complex correlated and organisation in order flow occurs in real markets.

8. CONCLUSION AND FUTURE WORK

In this paper we developed a zero-intelligence model of financial markets based on an empirical analysis of the Paris Bourse exchange [1]. We used this model to test whether a "zero-intelligence plus" model could simultaneously explain several of the stylized facts of financial time series data by systematically evaluating seven different experimental treatments in which we incrementally added minimalist levels of strategic (conditional) behaviour to our model. We tested the resulting time series of returns for the following statistical features: fat tails, volatility clustering, high-frequency persistence and non-Gaussianity. Surprisingly, the "zero-intelligence plus" treatments do *not* introduce more realism to the time series.

Our new zero-intelligence model therefore highlights the fact there are many ways to simulate random order flow and within each structure there is implicit intelligence. In our model agents specifically remove or add liquidity to the market as the random order book events are classified with implicit knowledge about price levels in the order book. In the [9], [11], [8] ZI models, it is left to the matching of the double auction to determine whether liquidity is removed or added changing price levels in the book despite the fact in a continuous double auction market with a visible order book this would be known to all agents. This motivates setting up a simulation model which takes into account more knowledge of the state of the book known in real time.

Using the decomposed microstructural events of [1] in our model, a new framework can be used to investigate adaptive order flow. Specifically it allows a more detailed understanding of the level of organised order flow amongst liquidity providers and takers is needed to understand real markets. We find changing the model parameters alters the degree of reality and introducing calibrated vectors improves results of stylised facts in terms of realism. Similar to other ZI models in the literature, the model manages to reproduce some realistic stylised facts, thus supporting the claim they are due to the microstructure and not strategic behaviour. In this case we manage to reproduce realistic Hurst exponents, fat tails and non-Gaussianity of returns. The model however in all treatments did not produce volatility clustering suggesting there is a more complex and correlated interplay of orderflow required for this phenomena. In the "cluster" treatment of our model a simple clustering of transactions was not sufficient to reproduce volatility clustering. Our results thus support the claim by [6] and [2] that along with ordering of transactions, there are also subtle fluctuations in the balance between liquidity taking and provision that acts as important factors in determining clustered volatility. This they argue would need to incorporate long-memory of supply and demand incorporating both order splitting and "stimulated refresh" liquidity provision. [14] discuss non-ZI explanations of volatility clustering suggesting social interactions amongst agents which employ herding behaviour (either by direct or indirect imitation) and/or order splitting of large trades are related to this phenomena.

The model could be extended with more complex dynamics, for example by using a reinforcement learning algorithm in which the propensity to choose a particular type of event adapts over time according to a profit signal such as the ef-

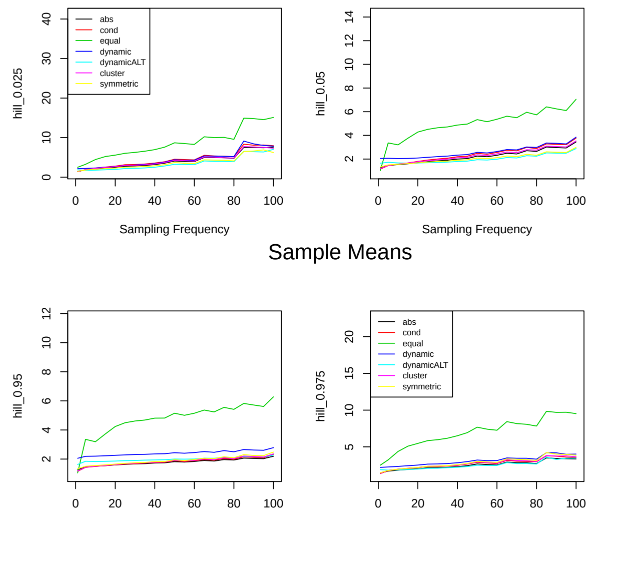


Figure 6: Sample means for the Hill Estimator of returns for all 7 treatments at the lower (0.025, 0.05) and upper (0.95, 0.975) quantiles.

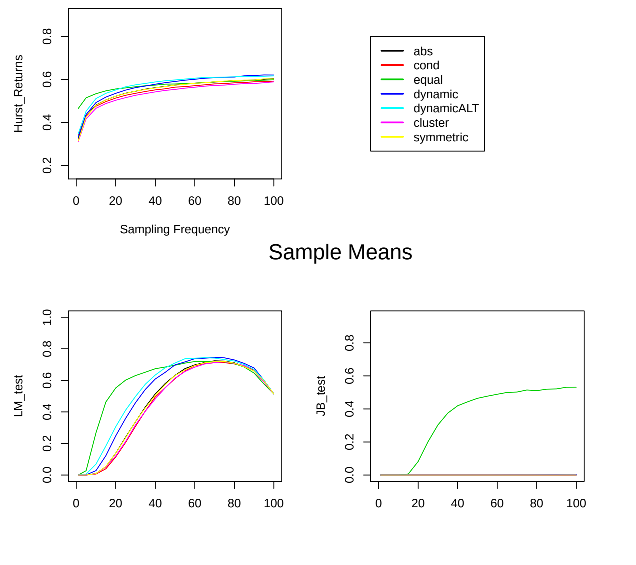


Figure 7: Sample means for the Hurst Exponent, ARCH-LM and Jacques-Bera test p-values of returns for all seven treatments.

fective cost of an order type. Such a learning model could possibly induce a more realistic "stimulated refresh" effect whereby clustered buying (selling) market order activity is followed by clustered selling (buying) limit order liquidity replenishing. Investigating this effect could provide an interesting avenue for future research to see its effect in reproducing realistic heavy tail and volatility clustering phenomena.

Our results provide further evidence that some phenomena in financial markets can be attributed soley to the market design, and do not require strategic behaviour. However, our findings also highlight the importance of time correlation of order flow; markets exhibit a complex interplay of latent supply and demand arising from strategic behaviour in which agents adapt their liquidity-taking and liquidity-providing behaviour depending on other agents' actions.

9. REFERENCES

[1] B. Biais, P. Hillion, and C. Spatt. An Empirical Analysis of the Limit Order Book and the Order Flow in the Paris Bourse. *The Journal of Finance*, 50(5):1655–1689, 1995.

[2] J. P. Bouchaud. The Endogenous Dynamics of Markets: Price Impact, Feedback Loops and Instabilities. In A. Berd, editor, *Lessons from the Credit Crisis*. Risk Publications, 2011.

[3] C. Chiarella and G. Iori. A simulation analysis of the microstructure of double auction markets. *Quantitative Finance*, 2(5):346–353, 2002.

[4] D. Cliff and J. Bruten. Minimal-Intelligence Agents for Bargaining Behaviors in Market-Based Environments. Technical report HPL-07-91, HP Labs, 1997.

[5] R. Cont. Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative Finance*, 1(2):223–236, Feb. 2001.

[6] L. Gillemot, J. D. Farmer, and F. Lillo. There's more to volatility than volume. *Quantitative Finance*, 6(5):371–384, 2006.

[7] D. K. Gode and S. Sunder. Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. *Journal of Political Economy*, 101(1):119–137, 1993.

[8] D. Ladley and K. R. Schenk-Hoppé. Do stylised facts of order book markets need strategic behaviour? *Journal of Economic Dynamics and Control*, 33(4):817–831, Apr. 2009.

[9] S. Maslov. Simple model of a limit order-driven market. *Physica A: Statistical Mechanics and its Applications*, 278(3-4):571–578, Apr. 2000.

[10] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, Jan. 1998.

[11] S. Mike and J. D. Farmer. An empirical behavioral model of liquidity and volatility. *Journal of Economic Dynamics and Control*, 32:200–234, 2008.

[12] T. Preis, S. Golke, W. Paul, and J. J. Schneider. Multi-agent-based Order Book Model of financial markets. *EPL (Europhysics Letters)*, 75(3):510–516, Aug. 2006.

[13] M. Wyart, J.-P. Bouchaud, J. Kockelkoren, M. Potters, and M. Vettorazzo. Relation between bid-ask spread, impact and volatility in order-driven markets. *Quantitative Finance*, 8(1):41–57, 2008.

[14] R. Yamamoto. Volatility clustering and herding agents: does it matter what they observe? *Journal of Economic Interaction and Coordination*, 6(1):41–59, May 2011.

A Scoring Rule-Based Mechanism for Aggregate Demand Prediction in the Smart Grid

Harry Rose, Alex Rogers, Enrico H. Gerding
Electronics and Computer Science, University of Southampton
{htrose,acr,eg}@ecs.soton.ac.uk

ABSTRACT

This paper presents a novel scoring rule-based strictly dominant incentive compatible mechanism that encourages agents to produce costly estimates of future events and report them truthfully to a centre. Whereas prior work has assumed a fixed budget for payment towards agents, this work makes use of prior information held by the centre and assumes a budget that is determined by the savings made through the use of the agents' information over the centre's own prior information. This mechanism is compared to a simple benchmark mechanism wherein the savings are divided equally among all home agents, and a cooperative solution wherein agents act to maximise social welfare. Empirical analysis is performed in which the mechanism is applied to a simulation of the smart grid whereby an aggregator agent must use home agents' information to optimally purchase electricity. It is shown that this mechanism achieves up to 77% of the social welfare achieved by the cooperative solution.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Economics, Theory

Keywords

Mechanism Design, Smart Grid, Information Aggregation, Scoring Rules

1. INTRODUCTION

There are numerous scenarios in which, in order for a centre to optimally perform a task, it must gather predictive information from other, potentially non-cooperative experts such that the centre can optimally plan for future events. In such situations, the centre often incurs a cost related to the imprecision of its predictions, and consequently, the ability to produce precise estimates results in a reduction of these costs. In non-cooperative situations, it becomes necessary for the centre to make payments to the experts in order to encourage them to report any relevant information that they have. When these experts are self-interested rational agents, the payments must be carefully designed in order to elicit the behaviour that the centre requires.

As an example, consider the instance of the above setting on which this paper focuses; information aggregation within the smart grid. In this scenario, an *aggregator agent* must purchase electricity for a set of homes, each of which is represented by its own *home agent*. Based on historical evidence, the aggregator has some belief of what each house will consume at a given future time, but each house has a wealth of information within it that can, *at a cost*, be collected and processed by the home agent in order to be used to make more precise estimates. It is the home agents' jobs to gather this information on behalf of the home owners and transmit it to the aggregator agent in the form of a probability distribution over the houses' possible consumptions at the specified time. The aggregator must pay a penalty for any difference between the amount it purchases and the amount its customers consume. Thus, with more precise information, the aggregator will in expectation make some savings. The aggregator then distributes a portion of these savings to the home agents as a reward for their information.

The truthful elicitation of information from self-interested agents has already been the subject of much attention in the literature. It has been shown that in scenarios where a number of expert agents can be called upon to make probabilistic estimates of some value, peer prediction techniques can be employed to ensure those agents report truthfully [4, 6]. Peer prediction rewards experts by fusing reports from a reference agent and the agent being scored; both of whom are making estimates of the same variable. After the value of the variable becomes known, the agent whose report is being evaluated is rewarded according to how its report affected the estimated likelihood of the realised event. This evaluation makes use of *proper scoring rules* – functions of probabilistic reports and outcomes that return a score that is only maximised in expectation when the agent reports truthfully. Indeed, scoring rules have also been the focus of much attention, since their original use in meteorology [1], through more general applications of evaluating predictors [8], and their more recent application in the field of prediction markets [7] wherein experts are asked by a centre to give a probabilistic response to some question, and also in the fair division of rewards among agents performing a task [2]. Moreover, strictly proper scoring rules have been generalised to take into account a prior distribution representing knowledge held by the centre [5].

However, there are four main limitations to these works

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

that limit their applicability to the problem discussed in this paper. Firstly, the works assume there exists multiple experts from which the centre can elicit information. In doing this, the experts' reports can be compared to each other, and the experts' payments based upon how correlated theirs and the other agents' reports were. This results in payments that are *Nash incentive compatible*, whereas a preferable solution concept would be *dominant strategy incentive compatible*, whereby truth telling is a dominant strategy regardless of the other agents' behaviour. Furthermore, in the scenario above, only a single agent exists per home, and agents are unable to measure their neighbours' demand. Consequently, agents' reports cannot be directly compared as the events they are predicting are unique to themselves. Secondly, in prior work, the reports from the experts are fused with one-another, whereas in the scenario above, the centre is interested in the *cumulative* demand of agents, and therefore must take the convolution of the reports. Thirdly, the solutions do not take into account prior knowledge held by the centre, with the exception of [5]. In the scenario above, the centre has prior information about each home, and will make savings based upon the precision of the information he uses. The mechanism should take this into account such that the aggregator does not pay for information less precise than his own, and does not run a deficit in paying for said information. The scoring rules in [5] have been adapted to take into account prior information. However, the computation of the scores can be problematic and are unbounded in the continuous domain. Finally, the solutions assume that the budget for payment to the agents is somehow fixed, whereas we propose to make use of the fact the centre has prior information, and base agents' rewards on the *savings made by the centre* in using the agents' reports over his own prior information.

Against this background, we develop a novel, scoring rule-based mechanism named the *sum of others' plus max* mechanism, which distributes payments to agents from a budget that is determined by their own reports in a way that is incentive compatible, and *ex ante* weakly budget balanced (i.e. the expected sum of payments to the agents is less than or equal to the budget allocated for rewards by the centre).

In more detail, this paper makes the following contributions to the state of the art:

- We present a new scoring rule-based mechanism named *sum of others' plus max* (SOM), which we apply to the scenario of information aggregation in the smart grid. The mechanism rewards agents *using a budget determined by their own reports*, and takes into account the agents' reports *and the centre's prior information.*
- We prove this mechanism to be *dominant strategy incentive compatible* and *ex ante weakly budget balanced.*
- We compare this mechanism using a computational approach to find equilibrium states to a benchmark mechanism in which rewards are divided uniformly between agents, and the cooperative social-welfare maximising solution. In doing so, we show that this mechanism is much more efficient than the benchmark, and obtains a social welfare that is up to 77% of that of the optimal cooperative solution.
- We show that SOM reduces the risk of the aggregator making a loss compared to the uniform mechanism.

The remainder of the paper is structured as follows: Section 2 presents a formal model of the information aggregation problem applied to the smart grid. Section 3 discusses two mechanisms to reward agents and their theoretical properties – the uniform mechanism, and the sum of others plus max mechanism. Section 4 then discusses a *cooperative* solution whereby all agents try to maximise the social welfare of the system. Section 5 uses empirical analysis to compare the two mechanisms and the social welfare solution. Finally, the paper concludes in Section 6.

2. THE INFORMATION AGGREGATION PROBLEM

This section presents a formulation of the information aggregation problem for demand prediction within the smart grid. In this scenario there are two types of agents – a single *aggregator agent*, and *n home agents*, $i \in N$, where $N = \{1, \cdots, n\}$. The aggregator's job is to gather information about the future electricity consumption of a set of homes and then buy electricity for those homes. The homes each have their own agent, whose job it is to collect specific, detailed consumption information about the home for which it is responsible and then to report it to the aggregator. The aggregator can then use this information to make better predictions of the future aggregate consumptions, which, due to the design of the electricity markets, reduces the total cost of the electricity consumed for all the homes.

In more detail, each day, D, is divided into a number of time periods, which, for simplicity and without loss of generality, we assume to be one. For each time period, the aggregator must purchase the amount of electricity it expects its agents to consume. The aggregator can purchase electricity in one of two markets, dependent on the time of purchase. Electricity can be bought one day ahead of its consumption in the *forward market*, in which case it costs f per unit of electricity. At the end of each day, the aggregator is charged for any imbalance between the amount it purchased in the forward market for consumption and the amount it actually consumed. We say these transactions are performed in the *balancing market* in which the prices are designed by the market regulator to penalise suppliers and consumers who do not generate or consume as they predicted. The price at which the grid buys back excess electricity, the system buy price, is $f - \delta^b$ per unit, and the cost per unit of electricity bought from the grid to fill any deficit, the system sell price, is $f + \delta^s$. Therefore, the total cost of consuming ω units of electricity when χ units are initially bought is given by:

$$\kappa(\omega\,|\chi) = f \cdot \chi + (\omega - \chi) \cdot \begin{cases} \left(f - \delta^b\right), & \chi > \omega \\ \left(f + \delta^s\right), & \chi < \omega \end{cases} \qquad (1)$$

Each agent, i, can generate *at a cost* an estimate, x_i, of its future consumption, represented by a Gaussian distribution[1] with mean, μ_i, and precision, $\theta_i = 1/\sigma_i^2$, such that $x_i = \langle \mu_i, \theta_i \rangle$. However, it is important to note that the mechanisms generalise to any probability distribution. The cost incurred by agent i when generating an estimate of pre-

[1]Gaussian distributions were chosen as the estimates produced by the agents are likely to be dependent on numerous noisy sources of information, and by the central limit theorem, the sum of noisy data results in a Gaussian error.

cision θ_i is:

$$C(\alpha_i, \theta_i) = \alpha_i \cdot \theta_i \tag{2}$$

where α_i is some positive, real-valued constant. The aggregator also maintains its own belief about what each agent i, will consume, $x_{a,i} = \langle \mu_{a,i}, \theta_{a,i} \rangle$. However, the aggregator does not incur a cost in maintaining its belief.

The day before the electricity is required, the aggregator asks each agent to report its estimate of tomorrow's consumption, $\hat{x}_i = \langle \hat{\mu}_i, \hat{\theta}_i \rangle$, as a Gaussian distribution with mean, $\hat{\mu}_i$, and precision, $\hat{\theta}_i$. The home agents are assumed to be strategic and they will try to maximise the benefit they receive; defined broadly as some payment for their report minus the cost of generating that report. As such, agents strategise over the precision of the estimate that they actually generate, $x_i = \langle \mu_i, \theta_i \rangle$, and also the mean and precision that they *report* to the aggregator, $\hat{x}_i = \langle \hat{\mu}_i, \hat{\theta}_i \rangle$. Consequently, an agent will misreport (i.e. $\hat{x}_i \neq x_i$) if it believes doing so will gain it a greater utility.

Once received, the estimates reported by the home agents are compared by the aggregator to its own information. The aggregator does not know the correlation between the reports of the agents and its own. Thus it takes the conservative action of assuming they are perfectly correlated, and simply takes the most precise estimate of its own and the agent's. In so doing, the aggregator produces the following aggregate belief vector:

$$\boldsymbol{x} = \langle x_1^*, \cdots, x_n^* \rangle \tag{3}$$

where,

$$x_i^* = \begin{cases} \hat{x}_i, & \text{if } \hat{\theta}_i > \theta_{a,i} \\ x_{a,i}, & \text{otherwise} \end{cases} \tag{4}$$

The result of Equation 4 is that the aggregator will only use the home agent's reported estimate, $\hat{x}_i$, if said estimate is more precise than the estimate the aggregator already has, $x_{a,i}$. Therefore, the aggregator will never use information that is less precise than its own belief. In certain situations, this can result in the aggregator losing information. However, in general this behaviour is a necessary consequence of the aggregator being unaware of the correlation of the information sources being received, although domain-specific knowledge might be applied to overcome this limitation. Since the aggregator is interested in the *cumulative* predicted demand of all homes, it convolves $\boldsymbol{x}$ to calculate a distribution that represents the expected total demand.

In order to make notation less verbose, let the expected total demand according to $\boldsymbol{x}$ be $\mu = \sum_{x_i^* \in \boldsymbol{x}} \mu_i^*$, and its precision $\theta = 1/(\sum_{x_i^* \in \boldsymbol{x}} 1/\theta_i^*)$. Similarly, for the aggregator's beliefs, $\boldsymbol{x}_a = \langle x_{a,1}, \cdots, x_{a,n} \rangle$, let the mean and precision be $\mu_a = \sum_{x_{a,i} \in \boldsymbol{x}_a} \mu_{a,i}$ and $\theta_a = 1/(\sum_{x_{a,i} \in \boldsymbol{x}} 1/\theta_{a,i})$.

Once the aggregator has collected estimates from all agents, it performs an optimisation to determine the amount of electricity it must purchase in the forward market such that its total expected cost is minimised. Essentially, the aggregator tries to minimise its expected loss in the balancing markets, which it does by solving the following equation:

$$\chi(\boldsymbol{x}) = \underset{z \in \Omega}{\arg\min} \; f \cdot z - \int_0^z (z-y)(f-\delta^b)\mathcal{N}(y;\mu,\theta)\ \mathrm{d}y + \int_z^\infty (y-z)(f+\delta^s)\mathcal{N}(y;\mu,\theta)\ \mathrm{d}y \tag{5}$$

At the end of each day, the actual amount consumed by each agent, defined by $\boldsymbol{\omega} = \langle \omega_1, \cdots, \omega_n \rangle$, becomes known to the aggregator. The total consumption is defined as $\omega = \sum_{\omega_i \in \boldsymbol{\omega}} \omega_i$. Each agent then pays the aggregator for the electricity their home consumed, at a rate of f_r per unit. The aggregator can also calculate the total cost it incurred through utilising the agents' estimates, $\kappa(\omega \,|\chi(\boldsymbol{x}))$, and the cost it *would* have incurred had it simply used its own prior information, $\kappa(\omega \,|\chi(\boldsymbol{x}_a))$.

The aggregator must then decide an amount to pay each home agent, P_i. Agent i's utility is then defined as follows:

$$U_i(\boldsymbol{x}, \boldsymbol{x}_a, \boldsymbol{\omega}) = P_i(\boldsymbol{x}, \boldsymbol{x}_a, \boldsymbol{\omega}) - C(\alpha_i, \theta_i) - \omega_i \cdot f_r$$

and the aggregator's utility is defined as:

$$U_a(\boldsymbol{x}, \boldsymbol{x}_a, \boldsymbol{\omega}) = \omega \cdot f_r - \kappa(\omega|\chi(\boldsymbol{x})) - \sum_{i \in N} P_i(\boldsymbol{x}, \boldsymbol{x}_a, \boldsymbol{\omega}) \tag{6}$$

Given that the agents are *rational*, they are able to strategise over their space of reports in order to determine how precisely that they generate their estimate and then, whether or not to truthfully report that estimate to the aggregator. Indeed, even after an agent has paid to produce an estimate, it might still gain a greater reward by misreporting. However, the aggregator is able to incentivise agents to behave in certain ways by carefully designing the reward function, P_i. In so doing, it wants to achieve two main goals: to incentivise home agents to make precise estimates and to incentivise agents to report those estimates truthfully. This is a problem that is discussed in the next section.

3. NON-COOPERATIVE MECHANISMS

This section presents two mechanisms that allocate rewards to agents for their information. A mechanism specifies a *transfer function*, which defines the reward an agent receives for a given reported estimate, $\hat{x}_i$, when an outcome, ω_i, is realised. Specifically, we consider three properties that are desirable in the scenario presented earlier. First, the mechanism should exhibit *individual rationality*. That is, in expectation, all agents gain a positive utility from participating in the mechanism. This is an essential requirement of any mechanism designed for use within an aggregation service to which customers may opt out – people will simply not use the service if they expect to be worse off by so doing. Second, the mechanism should be *incentive compatible*, which means that an agent maximises its expected utility by truthfully reporting its estimate. This has obvious advantages in the aggregation scenario described earlier – the aggregator needs to know the real estimates the agents hold in order to generate an accurate estimate of their aggregate future consumption. Third, it should be *budget balanced*, which states that the aggregator does not run into deficit after paying the agents for their estimates. We consider a mechanism to be budget balanced if the aggregator spends equal to or less than it would have, had it only used its own estimates and not elicited estimates from the home agents.

Given this, the next section discusses how the savings are calculated in order to determine the aggregator's budget. Afterwards, the mechanisms that distribute this budget are discussed. First to be discussed is the *uniform mechanism*; a simple mechanism whereby the savings made by the aggregator are equally divided amongst the home agents. Next, a further mechanism named *sum of others' plus max* is discussed, which uses the spherical scoring rule in order to define the proportion of the savings distributed to each agent.

3.1 Calculation of Savings

The budget the aggregator uses to reward the home agents in the mechanisms presented here is a function of the total savings made by the aggregator buying electricity using the home agents' information over its own (if the home agents' information is more precise than the aggregator's). Formally, when the agents consume $\boldsymbol{\omega}$, their aggregated reports are $\boldsymbol{x}$, and the aggregator's aggregated prior information is $\boldsymbol{x}_a$, the savings made by the aggregator are:

$$\Delta\left(\boldsymbol{x}, \boldsymbol{x}_a, \boldsymbol{\omega}\right) = \kappa\left(\omega \left| \chi\left(\boldsymbol{x}_a\right)\right.\right) - \kappa\left(\omega \left| \chi\left(\boldsymbol{x}\right)\right.\right) \tag{7}$$

The aggregator may not necessarily decide to allocate the whole amount of savings to the agents' reward budget. Instead, it allocates a fraction $0 \leq \lambda \leq 1$ to distribute, thereby guaranteeing the aggregator a certain fraction of the savings. However, the aggregator is still left with the problem of allocating the λ savings to the agents such they are incentivised to report precise estimates truthfully. The next sections are devoted to describing two mechanisms. First, their formal properties are discussed, then further examination of their properties is performed using empirical evaluation.

3.2 Uniform Mechanism

The simplest mechanism presented in this paper simply divides the savings made by the aggregator equally amongst the agents. In this case, the reward given to each agent is:

$$P_i^{\mathbf{U}}\left(\boldsymbol{x}, \boldsymbol{x}_a, \boldsymbol{\omega}, n\right) = \frac{1}{n} \cdot \lambda \cdot \Delta\left(\boldsymbol{x}, \boldsymbol{x}_a, \boldsymbol{\omega}\right)$$

It is clear to see that the uniform mechanism is budget balanced (i.e. it always distributes 100% of it's allocated budget) – the budget is equally split into n amounts, which are then awarded to n agents, thus always distributing 100% of the allocated budget. Further to this, Theorem 3.1 provides a proof that shows the uniform mechanism to be Nash incentive compatible (that is, reporting truthfully is a Nash equilibrium). Thus, under this mechanism, when all agents are truthful, no single agent has incentive to misreport.

THEOREM 3.1. *The uniform mechanism is Nash incentive compatible, i.e. truth telling is a Nash equilibrium.*

PROOF. The aggregator buys an amount of electricity for the agents that minimises the total expected cost based on the agents' reported estimate. Clearly if all agents report truthfully, one agent deviating will only cause a larger error between the amount consumed and purchased, resulting in less savings to be distributed to the agents and therefore a lower utility for that agent. Therefore, when all agents report truthfully, a single agent is unable to improve its expected utility by misreporting. However, note that truth telling is not a dominant strategy; if an agent knows its neighbour will misreport, the agent can obtain a better expected reward by also misreporting such that its error cancels out the error made by the neighbour. □

Using this mechanism, all agents are rewarded equally irrespective of their actual contribution. An ideal mechanism would reward the agents more *fairly*, by making greater payments to those agents whose estimates made the most significant increase in the aggregator's savings. Furthermore, the fact that truth telling is only a Nash equilibrium means that agents can potentially expect to benefit from misreporting their estimates if they believe other agents will do the same. Therefore, a better solution is a mechanism that is dominant strategy incentive compatible. That is, a mechanism where an agent's utility is maximised when reporting truthfully *regardless* of its belief of the other agents' actions. With this in mind, we discuss next the *sum of others' plus max* mechanism (SOM), which uses strictly proper scoring rules in order to achieve dominant strategy incentive compatibility. Strictly proper scoring rules are functions that take a probabilistic estimate reported by an agent, and an outcome. Their expected value is maximised only when an agent truthfully reports their estimates. SOM uses the *spherical scoring rule*, which is discussed in the next section.

3.3 Spherical Scoring Rule

The mechanism in the next section is based on the spherical scoring rule. For a given prediction of an event with mean $\hat{\mu}_i$, and precision $\hat{\theta}_i$, and a realisation of that event, ω_i, the spherical rule is defined as follows:

$$S\left(\omega_i; \hat{\mu}_i, \hat{\theta}_i\right) = \frac{\mathcal{N}\left(\omega_i; \hat{\mu}_i, \hat{\theta}_i\right)}{\sqrt{\int_{-\infty}^{\infty} \mathcal{N}\left(x; \hat{\mu}_i, \hat{\theta}_i\right)^2 \, \mathrm{d}x}} \tag{8}$$

The spherical rule is one of three *strictly proper* scoring rules often studied in literature – the other two being the logarithmic, and quadratic scoring rules. The term strictly proper means that the expected score awarded by the function is maximised exclusively when the agent truthfully reports its estimate. The spherical rule was chosen over the much simpler logarithmic rule because it has a strict lower bound of 0, whereas the logarithmic rule is unbounded. As a result, the use of the logarithmic scoring rule could theoretically end with a customer becoming forever in debt to the aggregation company after having received a score of $-\infty$. This is clearly unsatisfactory, and it could be argued that this fact, no matter how rare its occurrence, could dissuade users from ever joining the aggregation service.

Using scoring rules to distribute payments to agents in continuous domains is non-trivial. This arises from the fact that $\mathcal{N}(\mu, \theta)$ is unbounded as $\theta \to \infty$. Clearly, if a home agent were to know exactly what it would consume in the next time period, paying it an infinite amount is not acceptable to the aggregator. Therefore, the score must be scaled in order to apply bounds. The next section discusses two methods for achieving this, and introduces the main contribution of this paper – the sum of others' plus max mechanism.

3.4 Sum of Others' plus Max

As has been discussed, scoring rules are used to assign scores to agents based on the accuracy and the precision of the estimates the agents report to the centre. Naturally, agents who report more precise estimates expect to get a higher score, and remembering that the cost agents incur when generating their estimates is proportional to the precision of their generated estimates, rewarding agents based on the score they achieve seems to be a natural development. Considering that we would also like to *fairly* distribute the savings to the agents in a way that is *budget balanced*, one method of distributing the savings to the home agents might be to scale the total savings by the fraction of the sum of all agents' scores that the agent had contributed. We call this mechanism the *percentage contribution* mechanism, where the reward each agent receives is given by:

$$P_i^{\mathbf{P}}\left(\boldsymbol{x}, \boldsymbol{x}_a, \boldsymbol{\omega}\right) = \frac{S\left(\omega_i; \hat{\mu}_i, \hat{\theta}_i\right)}{\sum_{x_j \in \boldsymbol{x}} S\left(\omega_j; \hat{\mu}_j, \hat{\theta}_j\right)} \lambda \cdot \Delta\left(\boldsymbol{x}, \boldsymbol{x}_a, \boldsymbol{\omega}\right)$$

However, despite the fact that this might be intuitively correct, the resulting payments are *not* incentive compatible. Agents are in fact able to *misreport* the precision of their belief in order to gain a higher reward by making their belief seem more precise. In this section, the percentage contribution mechanism is adapted to make the sum of others' plus max (SOM) mechanism, which *is* incentive compatible, but is only *ex ante* weakly budget balanced. That is, in expectation, SOM distributes *at most* 100% of its budget.

This mechanism takes into account not only the spherical score (defined in Equation 8) achieved by the agent, but also those achieved by the other agents in the system. Payments are then determined by multiplying those scores by the savings made by the aggregator when using the reports from the *other* agents in the system, and the aggregator's prior knowledge in place of the report from the agent who is being rewarded. This is necessary in order to preserve incentive compatibility. Furthermore, to ensure agent's payments never outweigh the savings made by the aggregator, it is necessary to provide an upper bound on the precision of reports accepted from agents, θ_{max}. If any agent reports a precision $\hat{\theta}_i > \theta_{max}$, their spherical score will be calculated as though $\hat{\theta}_i = \theta_{max}$. Formally, the payment agent i obtains, given all agents' estimates is given by:

$$P_i^{\mathbf{S}}\left(\boldsymbol{x}, \boldsymbol{x}_a, \boldsymbol{\omega}\right) = \frac{S\left(\omega_i; \hat{\mu}_i, \hat{\theta}_i\right) \cdot \lambda \cdot \Delta\left(\boldsymbol{x}_{-i} \cup \{x_{a,i}\}, \boldsymbol{x}_a, \boldsymbol{\omega}\right)}{S\left(\omega_i; \omega_i, \theta_{max}\right) + \sum_{x_j \in \boldsymbol{x}_{-i}} S\left(\omega_j; \hat{\mu}_j, \hat{\theta}_j\right)}$$

where $\boldsymbol{x}_{-i} = \boldsymbol{x} \setminus \{x_i^*\}$, and the term, $S\left(\omega_i; \omega_i, \theta_{max}\right)$, represents the maximum score that can be achieved by an agent – the score achieved when reporting the maximum possible precision, θ_{max}, and reporting an estimate with mean ω_i when ω_i actually does occur. It can be seen that by using only the savings made by the other agents, the only term that is dependent on the report from the agent being rewarded is the scoring rule. Moreover, the spherical scoring rule was specifically chosen due to it's strict propriety, and therefore these payments are incentive compatible, as is shown in Theorem 3.2.

Theorem 3.2. *SOM is dominant strategy incentive compatible, i.e. truth telling is a strictly dominant strategy.*

Proof. The maximum score is a constant value set by the mechanism designer. Furthermore, the agent's report is excluded from the calculation of the savings made. Ergo the agent is unable to affect the savings used to calculate its payment. Thus, the savings made by the other agents are in effect a constant. Given that, SOM is simply an affine transformation of the spherical scoring rule, which maintains strict propriety and therefore incentive compatibility. The fact that the score is *strictly* proper means that the expected score is a unique maximum when an agent reports truthfully. Thus, the expected reward an agent receives is also a unique maximum when it reports truthfully. Therefore, the mechanism is strictly dominant incentive compatible. $\square$

In addition to truth-telling being a strictly dominant strategy, the rewards to agents made by this rule are fairer than those of the uniform mechanism in that the agents are directly compared with each other based upon their score. The reward is simply a scaled fraction of the agents' spherical score over the sum of all other agents' scores. Thus, if an agent scores highly because it reported a precise estimate, and the other agents score lower because of imprecise reports, the first agent will receive a greater share of the savings made. It is essential to divide the agent's spherical score by the sum of the other agents' prescaled scores *plus the maximum score* in order to maintain weak budget balance. Theorem 3.3 provides a proof of the fact that SOM is ex ante weakly budget balanced.

Theorem 3.3. *SOM is ex ante weakly budget balanced.*

Proof. Let each agent, i, obtain the score S_i, and $\bar{\Delta}(\theta)$ be the expected savings made when the agents' reports produce an aggregate precision of θ. In SOM when each agent, i's, report has precision θ_i, the total expected payout is:

$$\sum_{\forall i \in N} \frac{S_i}{S_{max} + \sum_{\forall j \in N \setminus \{i\}} S_j} \cdot \bar{\Delta}\left(\left(\sum_{\forall j \in N \setminus \{i\}} \frac{1}{\theta_j} + \frac{1}{\theta_{a,i}}\right)^{-1}\right)$$

and the aggregator's total expected savings is

$$\bar{\Delta}\left(\left(\sum_{\forall i \in N} \frac{1}{\theta_i}\right)^{-1}\right)$$

The sum of the fraction of scores is ≤ 1, and $\bar{\Delta}(\theta)$ is strictly increasing with θ. Therefore, it is sufficient to prove:

$$\left(\sum_{\forall j \in N \setminus \{i\}} \frac{1}{\theta_j} + \frac{1}{\theta_{a,i}}\right)^{-1} \leq \left(\sum_{\forall i \in N} \frac{1}{\theta_i}\right)^{-1} \quad \forall i \in N$$

We start with the axiom,

$$\theta_{a,i} \leq \theta_i$$

Adding $\theta_i \theta_{a,i} \gamma > 0$ to both sides gives:

$$\theta_{a,i} + \theta_i \theta_{a,i} \gamma \leq \theta_i + \theta_i \theta_{a,i} \gamma$$

Which factorises to give:

$$\theta_{a,i}\left(\theta_i \gamma + 1\right) \leq \theta_i\left(\theta_{a,i} \gamma + 1\right)$$

The bracketed expressions are strictly positive. Therefore, it can be simplified to give:

$$\left(\gamma + \frac{1}{\theta_{a,i}}\right)^{-1} \leq \left(\gamma + \frac{1}{\theta_i}\right)^{-1}$$

Substituting γ for $\sum_{\forall j \in N \setminus \{i\}} \frac{1}{\theta_j}$, we are left with

$$\left(\sum_{\forall j \in N \setminus \{i\}} \frac{1}{\theta_j} + \frac{1}{\theta_{a,i}}\right)^{-1} \leq \left(\sum_{\forall i \in N} \frac{1}{\theta_i}\right)^{-1}$$

That is, the precision of the aggregate report made by the aggregator when using its own information in place of agent i's is less than the precision of using all agents reports, when each agent reports a precision greater than or equal to the aggregator's precision. Combined with the fact that the expected savings are strictly increasing with precision, and the sum of fractions of the budget allocated to each agent is less than or equal to one, this shows the sum of others plus max mechanism is *ex ante weakly* budget balanced. $\square$

There are numerous advantages to using SOM over the simple uniform mechanism presented earlier. Firstly, truth telling strictly dominates all other strategies. As a result, reporting truthfully will *always* maximise the agent's expected reward, regardless of the other agents' actions. This is not the case in the uniform mechanism wherein truth telling is only a Nash equilibrium. For example, if an agent were to learn that its neighbour were to misreport its estimate, it too could misreport in order to offset the other agent. However, a disadvantage of SOM compared to the uniform mechanism is that it is only ex ante weakly budget balanced – a weaker concept than the strict budget balance exhibited by the uniform mechanism. The home agents might make small losses when the other agents' predictions are poor. However, in expectation, home agents' utilities will always be positive as they are able to strategise over the precision the generate in order to maximise their utility. This is further explained with the aid of empirical evidence in Section 5.3.

4. THE SOCIAL WELFARE SOLUTION

While the design of the two previous mechanisms assumes that the agents are non-cooperative – that is, they seek only to maximise their own profit – the social welfare solution assumes *cooperation* between the agents. In the social welfare solution, agents ignore their own reward and instead maximise the sum of all agents' utilities within the system. In the case shown in this paper, when maximising social welfare, each agent maximises the following function:

$$U^*(\boldsymbol{x}) = \int \cdots \int_0^\infty \Delta(\boldsymbol{x}, \boldsymbol{x}_a, \boldsymbol{\omega}) - \sum_{\forall i \in N} C(\alpha_i, \theta_i) \; \mathrm{d}\omega_1, \cdots, \mathrm{d}\omega_n \tag{9}$$

The social welfare solution is significant as it provides an upper bound for the social welfare that can be achieved within the system. This result can then be used in order to ascertain the *efficiency* of any mechanism that works in the scenario, where more efficient mechanisms are deemed to be those whose social welfare is closer to that of the maximum social welfare. We use the social welfare result in the next section in order to analyse the efficiency of the sum of others plus max, and uniform mechanisms, and to discover the properties that arise in a cooperative model.

The previous sections have introduced two new mechanisms – the *uniform* mechanism and *sum of others' plus max* – and in doing so have discussed and proved some of the formal properties of said mechanisms. The solution whereby agents act cooperatively to maximise social welfare has also been introduced. The next section uses empirical analysis to further analyse emergent behaviour under equilibrium.

5. EMPIRICAL EVALUATION OF THE MECHANISMS

Given the theoretical properties of the mechanisms discussed in the previous sections, it is clear that agents will truthfully report their information to the aggregator. However, although rational agents will be truthful, they will strategise over the *actual* generated precision of their estimate. Therefore, to further analyse the properties of the mechanisms, we must solve for the equilibrium precision of the agents.

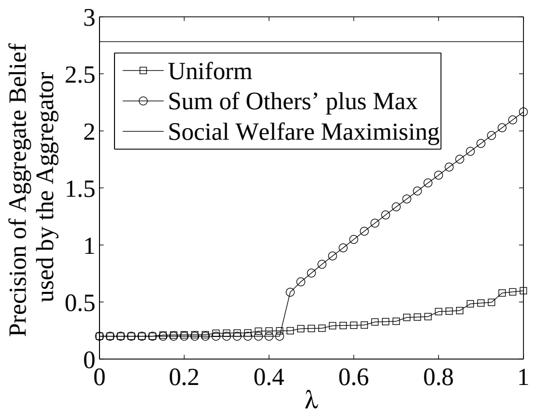


Figure 1: The aggregated precision of the agents' reports, versus the fraction of the savings to be distributed by the aggregator, λ.

Unfortunately, this equilibrium cannot be calculated, and thus, iterated best response is used to computationally find the home agents' equilibrium strategies (the precisions of their generated reports). To aid convergence to an equilibrium, a dynamic named partial best response is used [3]. Equilibrium is detected by measuring the variance of the agents' chosen strategies, and the algorithm is deemed to have reached an equilibrium point when the variance of the last 10 strategies chosen by each agent falls below a predetermined threshold (10^{-8} in our experiments).

Once the equilibria are found, the expected savings and the agents' expected rewards and utilities are analytically calculated. Market values are set to $f = 100$, $\delta^b = 50$, and $\delta^s = 70$. Agents' costs are set to $0.01, 0.02, \ldots, 0.1$, and kept constant throughout each simulation. The aggregator is assumed to have prior information that allows it to make estimates of each houses' demand with precision $\theta_a = 2$.

The remainder of this section discusses the results from these simulations. In Section 5.1, the aggregate precision of all agents' reports are compared for the two mechanisms and the social welfare solution. Next, Section 5.2 discusses the efficiency of each mechanism, where efficiency is defined in terms of a percentage of the social welfare achieved in the cooperative solution. Finally, Section 5.3 discusses the risks to the aggregator in using each mechanism.

5.1 Precision

In this section, the aggregated precisions of the agents' reports under the two mechanisms and the social welfare maximising case are analysed. Figure 1 shows the aggregated precision of all agents' reports against the fraction of the savings the aggregator allocates for distribution. The fact that the agents are playing unique equilibrium strategies that maximise their utility functions means that there is no error in the agents' chosen strategies, and consequently no error in the aggregated precision of the agents' reports. Thus, error bars have been omitted from the plot in Figure 1.

It can be seen from Figure 1 that in terms of aggregated precision, SOM vastly outperforms the uniform mechanism for values of $\lambda > 0.425$, obtaining an aggregated precision which is up to four times the aggregated precision obtained by the uniform mechanism. The step change in the SOM precisions between $\lambda = 0.4$ and $\lambda = 0.5$ occurs because lower values of λ are not incentivising the agents' to produce estimates as their additional reward is outweighed by their costs. Furthermore, steps can be seen in the uniform plot as each agent becomes incentivised to produce a report. The baseline at $\theta = 0.2$ occurs when no agents are incentivised to produce a report with $\theta_i > \theta_{a,i}$. Consequently, the aggregated belief consists only of the aggregator's prior knowledge, thus $\theta = \theta_a/n$ assuming $\theta_a = \theta_{a,i}$, $\forall i \in N$.

Figure 1 also shows the aggregated precision of the reports made by the home agents under a cooperative, social welfare maximising setting. In this setting, each agent's strategy is constant with respect to λ as the agents do not take into account their own reward, as is shown in Equation 9.

Furthermore, note that it is not the absolute value of the reward that is important in determining the strategy to be adopted by the agents, but the gradient of the reward function with respect to the agents' precision, θ_i. The absolute value only becomes relevant if the agent has the choice between the mechanism that is in use. Agents will choose a precision at which the gradient of the reward function, $dP/d\theta_i$ equals the gradient of their cost function $dC_i/d\theta_i$, or α_i. Therefore, it may well be possible to construct a reward function that encourages agents to generate yet more precise estimates than they do with SOM. However, as we discuss in the next section, SOM is already up to 77% efficient, and therefore any increase in the precision generated by agents will only produce marginally improved social welfare.

5.2 Efficiency

The mechanisms discussed in this paper do not 'burn' any unallocated savings. That is, savings that are not distributed to the home agents as payment are not simply discarded. Instead, any unallocated savings are returned to the aggregator to add to its profit. In this way, the mechanisms always allocate 100% of their budget. However, under this model, the budget for each mechanism is itself dependent on the actions of the home agents. Consequently, each mechanism will still result in a different social welfare. With this in mind, in this paper, efficiency is defined as the social welfare achieved by the mechanism expressed as a percentage of the social welfare that is achieved when agents act cooperatively to maximise additional social welfare.

Figure 2 shows the sum of the home agents' additional utilities against the additional utility gained by the aggregator in using each mechanism compared to using no mechanism for values of λ between 0.0 and 1.0. As before, the agents' behaviour in the social welfare solution is independent of λ. It can be seen that SOM is more efficient than the uniform mechanism in that it has points closer to the social welfare maximising solution. Furthermore, for any utility received by the home agents, the aggregator's additional utility is greater under SOM than the uniform mechanism.

Having discussed the expected utilities that are obtained by the agents and aggregator, we see that the aggregator always expects to receive a greater utility when using SOM over the uniform mechanism. However, observing samples of individual rounds shows that there are occasions wherein the aggregator makes a loss. The next section analyses the risk

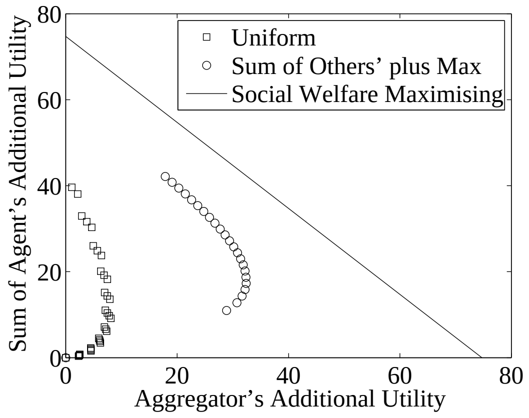


Figure 2: The additional utility of the aggregator when using a mechanism as opposed to using no mechanism against the sum of agents' additional utilities.

to the aggregator in using SOM compared to the benchmark.

5.3 Risk for the Aggregator

This section discusses the risk to the aggregator when using SOM and the uniform mechanism compared to using no mechanism at all – i.e. the aggregator simply using its own beliefs. There is always an element of risk for the aggregator, which arises due to errors made when predicting the demand of the home agents. It should be noted that this work assumes that the agents' beliefs are on average *accurate*. Thus, on average, the savings made in using the agents' more precise information is positive. Nevertheless, it is possible for savings on occasion to be negative, when an agent is confident in its belief, but is incorrect.

However, there are a number of ways the aggregator can mitigate such risk. For one, simply increasing the number of agents being aggregated over decreases the risk to the aggregator, as can be seen by the 'no mechanism' line in Figure 3. The use of mechanisms further reduces the aggregator's risk by encouraging agents to produce more precise estimates.

The results shown in this section use the same experimental setup as above, with the retail price, arbitrarily set to $f_r = f$ (note in a real situation, the aggregator would set $f_r > f$ in order to gain some profit). In addition, the value at risk is calculated through repeated simulation. In more detail, once the agents' strategies have been determined, the simulation of the aggregator purchasing electricity using the agents' reports runs in full for 1000 rounds. In each round, each house, i, is assigned a total consumption, ω_i, sampled from a uniform distribution over the range $[30, 50]$. The agent's own estimate of its consumption is then sampled from a normal distribution around its consumption such that $\mu_i \sim \mathcal{N}(\omega_i, \theta_i)$, where θ_i is the precision chosen by agent i in the equilibrium found through iterated best response as described earlier. The same procedure is used to generate the aggregator's belief using θ_a. The aggregator pays each agent their reward in accordance with the model in Section 2 and mechanisms in Section 3. After each round, the aggregator's utility is calculated (Equation 6). The value of the aggregator's utility at 5% risk is calculated by taking the aggregator's fifth percentile utility from the 1000 rounds. This process is repeated 20 times, and the mean and standard

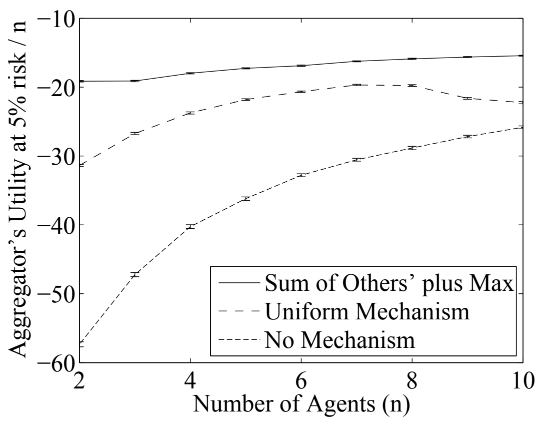


Figure 3: Value of the aggregator's utility per agent at 5% risk for two to ten agents when $\lambda = 0.5$.

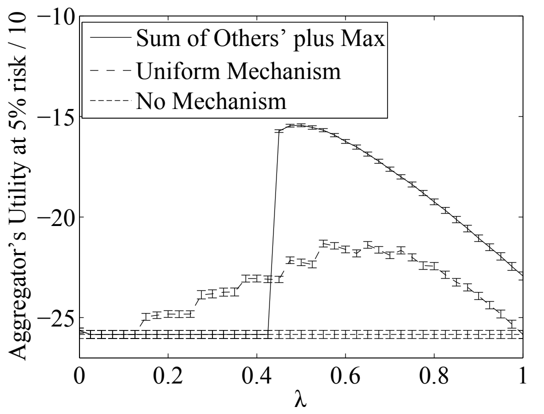


Figure 4: Value of aggregator's utility per agent at 5% risk for ten agents and λ between 0 and 1.

error of the value at 5% risk are taken.

Risk can be further mitigated by the aggregator by adjusting λ, as shown in Figure 4. It can be seen that SOM results in a greatly reduced risk to the aggregator for values of $\lambda > 0.425$ resulting from the high precision estimates that SOM encourages agents to produce. For higher values of λ, although the agents are still producing high precision estimates, as shown in Figure 1, the aggregator is giving away a larger portion of the savings to the agents, thereby decreasing its utility. For values of $\lambda < 0.425$, as shown in Figure 1, agents are not incentivised to produce reports, and thus the risk to the aggregator is equal to the risk when no mechanism is employed. The step changes in the 'uniform' line of Figure 4 arise due to agents individually becoming incentivised to produce estimates for the aggregator.

An additional source of risk comes from the *ex ante* weakly budget balanced nature of SOM; the mechanism is budget balanced *in expectation*, but there may be instances wherein the aggregator makes a loss. Furthermore, in situations whereby the savings made by the aggregator are negative – i.e. the agent's estimates were less accurate than the aggregator's prior estimate – the fact that the mechanism is ex ante, *weakly* budget balanced, means the aggregator will, in expectation, regain *at most* λ of the loss made.

However, it is worth noting that at no point in Figures 3 and 4, is the aggregator worse off by using either mechanism compared to using no mechanism at all. Furthermore, SOM, for $\lambda > 0.425$, maximises the utility of the aggregator. Thus a risk neutral aggregator, who is able to strategise over the mechanism used, will always choose SOM, with $\lambda > 0.425$.

6. CONCLUSIONS

This paper discussed mechanism design in scenarios wherein a centre has some imprecise information regarding a set of values that, when aggregated, provide information it must use to optimally procure goods at a cost. Each variable has a single expert agent that is able, at a cost, to report to the centre more precise information regarding the value. Additionally, the expected cost incurred by the centre increases with the precision of the information it uses to procure said goods. A dominant strategy incentive compatible scoring rule-based mechanism named *sum of others' plus max* (SOM) was developed, which rewards agents from a budget that is equal to the savings made by the centre in using the agents' information over its own.

SOM was compared to a simple mechanism whereby the agents are paid by dividing the savings made equally among the agents. It was shown that the sum of others plus max mechanism increased social welfare compared to the uniform mechanism, and that the social welfare achieved by the sum of others plus max mechanism was 77% that of the optimal solution. Empirical evidence was provided that shows that SOM reduces the risk to the aggregator compared to using a simple uniform mechanism or no mechanism at all.

Future work will investigate the combination of the sum of others plus max mechanism with additional incentives. For example, in the smart grid, demand smoothing is desirable as it reduces the need for extraneous and costly standby generation capacity on the grid. An additional incentive to build into this mechanism would therefore be to reduce the variance of the realised consumptions of each house, ω_i, as agents can potentially gain better payoffs by making their consumptions unpredictable to the aggregator.

7. REFERENCES

[1] G. Brier. Verification of forecasts in terms of probability. *Monthly Weather Review*, 78(1):1—3, 1950.

[2] A. Carvalho and K. Larson. A truth serum for sharing rewards. In *Proc. of AAMAS '11, Taipei, Taiwan*, pages 635—642, 2011.

[3] D. Fudenberg and D. K. Levine. *The Theory of Learning in Games (Economic Learning and Social Evolution)*. The MIT Press, 1998.

[4] N. Miller, P. Resnick, and R. Zeckhauser. Eliciting informative feedback: The peer-prediction method. *Management Science*, 51(9):1359—1373, 2005.

[5] R. Nau, V. R. Jose, and R. Winkler. Scoring rules, entropy, and imprecise probabilities. In *Proc. of ISIPTA '07*, pages 307—315, 2007.

[6] A. Papakonstantinou, A. Rogers, E. H. Gerding, and N. R. Jennings. Mechanism design for the truthful elicitation of costly probabilistic estimates in distributed information systems. *Artificial Intelligence*, 175(2):648 – 672, 2011.

[7] P. Shi, V. Conitzer, and M. Guo. Prediction mechanisms that do not incentivize undesirable actions. In *Proc. of WINE '09*, pages 89—100, Berlin, Heidelberg, 2009.

[8] R. Winkler. The quantification of judgment: Some methodological suggestions. *Journal of the American Statistical Association*, 62(320):1105—1120, 1967.

A Model-Based Online Mechanism with Pre-Commitment and its Application to Electric Vehicle Charging

Sebastian Stein, Enrico Gerding, Valentin Robu and Nicholas R. Jennings
Electronics and Computer Science, University of Southampton
Southampton, SO17 1AE, United Kingdom
{ss2,eg,vr2,nrj}@ecs.soton.ac.uk

ABSTRACT

We introduce a novel online mechanism that schedules the allocation of an expiring and continuously-produced resource to self-interested agents with private preferences. A key application of our mechanism is the charging of pure electric vehicles, where owners arrive dynamically over time, and each owner requires a minimum amount of charge by its departure to complete its next trip. To truthfully elicit the agents' preferences in this setting, we introduce the new concept of pre-commitment: Whenever an agent is selected, our mechanism pre-commits to charging the vehicle by its reported departure time, but maintains flexibility about *when* the charging takes place and at *what rate*. Furthermore, to make effective allocation decisions we use a model-based approach by modifying Consensus, a well-known online optimisation algorithm. We show that our pre-commitment mechanism with modified Consensus incentivises truthful reporting. Furthermore, through simulations based on real-world data, we show empirically that the average utility achieved by our mechanism is 93% or more of the offline optimal.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*multiagent systems*

General Terms

Algorithms, Design, Economics

Keywords

electric vehicles, mechanism design, pricing

1. INTRODUCTION

Recent years have seen a proliferation of interest in electric vehicles (EVs), generally perceived as a key technology for achieving sustainable mass transportation with low carbon emissions [7]. While EVs are a promising technology, their widespread use is also expected to place considerable strains on existing electricity distribution networks. EVs typically require high charging rates, up to 3 times the maximum demand of a typical home. This means that if all vehicle owners plug in at peak times (typically in the early evening), the transformers involved in distributing electricity to local neighbourhoods may be overloaded by the additional demand [7]. To this end, the charging of vehicles needs to be scheduled in order to balance the load. However, different consumers may have different time constraints and willingness to pay, which the schedule needs to take into consideration. To address this challenge, in this paper, we apply and extend techniques from mechanism design and the stochastic optimisation literature to ensure incentive compatibility (i.e., to incentivise *strategic* agents to reveal their preferences truthfully), and produce effective schedules for EV charging.

Dealing with the limitation of local distribution networks has been discussed in a range of recent works. For example, [2] and [12] provide a thorough descriptive analysis of this problem, although they only discuss the problem at a high level and do not propose specific scheduling heuristics. Moreover, these and most other papers assume that information about the EVs is known and they do not consider the elicitation problem, where strategic agents may misreport their preferences if this is in their best interest.

Framed in more general terms, the problem involves the real-time scheduling of jobs released over time (in our case, electric vehicles that require a certain amount of charge by their departure) sharing a scarce resource (in our case, electricity that is limited by the maximum transformer capacity), and given uncertainty about future arrivals. Such problems are addressed in the important and growing field of stochastic optimisation [6]. However, it has been shown that solving these problems optimally is NP-hard (see [10] for an overview) and many heuristics have been developed. The approach in our paper is based on one of the most widely used such heuristics, the Consensus approach introduced in [1]. In their approach, a number of future scenarios are sampled, and then the scheduling is solved for each of these scenarios (which can be solved using an offline algorithm). Then, the decision whether to schedule a particular job (in our case, an EV) is made based on a Consensus vote between these scenarios. However, unlike our work, [1] only applies to settings with a single machine (corresponding to a setting where only a single EV can be charged at any time) and agents are assumed to be non-strategic.

A number of papers have considered scheduling with strategic agents. Specifically, [11] examines the scheduling of jobs on a single machine and proposes an incentive compatible mechanism for this setting. However, their work assumes a computational setting where the results of a job are released to the agents only on completion or by the agent's reported deadline. This approach cannot be used for electricity distribution, since electricity must be allocated instantly when available. More recently, [5] proposes a mechanism that deals with multi-dimensional, marginally decreasing valuations, showing how this setting naturally applies to the charging of *hybrid* EVs (where any shortfall in electricity can be supplemented by fuel). However, that approach does not readily extend to the case of pure EVs, which is better characterised by complemen-

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

tary preferences (since agents only receive value when the battery is sufficiently charged), and it assumes both discrete time and supply. Furthermore, all of these approaches are *model-free*, i.e., they do not consider possible future arrivals in the allocations. This can be especially inefficient in case of complementary preferences.

There are a few online mechanism design papers which also account for future arrivals [8, 3]. Like our paper, these are based on a version of the Consensus algorithm, but their approach results in allocations which are not necessarily *monotonic* (a necessary condition for incentive compatibility). As a result, they require an additional *ironing* procedure which cancels allocations that violate the monotonicity property. This approach has two drawbacks. First, ironing is computationally prohibitive and thus impractical for the EV domain, since it needs to consider all possible misreports in order to establish whether cancelling an allocation is required. Second, it causes inefficiencies in the allocation, since ironed resources are lost and cannot be allocated to any other agents. Furthermore, unlike our setting, both [8] and [3] assume discrete time and indivisible resources.

In this work, we address these limitations by proposing the first incentive-compatible, model-based mechanism for a real-time setting with a continuously divisible resource. This achieves incentive compatibility by ensuring monotonicity for all allocations, thus avoiding the need for expensive ironing. In more detail, we make the following contributions to the state of the art:

1) We introduce, for the first time, the notion of *pre-commitment* in online mechanism design. When the mechanism decides to pre-commit to an agent, it ensures that sufficient resources are reserved for the agent, but, importantly, retains flexibility over when and how the resource is allocated. This leaves considerable flexibility for accommodating other, potentially less patient agents in the future.

2) To ensure incentive compatibility, we furthermore modify the existing Consensus algorithm. Specifically, we introduce a *serialisation* of agents, ensuring that an unallocated agent cannot influence future pre-commitment decisions. Second, we ensure agents do not have an incentive to delay their arrival in the system by identifying *re-evaluation points*, at which scheduling and pre-commitment decisions may change, and by using *partly-fixed* schedules, whereby the schedule for committed agents is fixed until the next re-evaluation point.

3) We evaluate our mechanism using real-world data from the largest trial of EVs in the UK and show that the average performance obtained by our mechanism is 93% or more of an offline optimal, and that it significantly outperforms a mechanism that allocates electricity without knowledge of the agents' preferences.

The remainder of the paper is organised as follows. In Section 2, we describe our model. In Section 3, we solve the online scheduling problem for a setting with cooperative agents, while in Section 4 we consider how to ensure truthful reporting by strategic agents. Finally, an experimental evaluation is presented in Section 5.

2. MODEL

We consider a setting where a limited and expiring resource (e.g., electricity) is continuously being produced and must be allocated to consumers immediately or is otherwise lost. Formally, the *production rate* (e.g., in kW) of the resource at time $t \in \mathbb{R}$ is given by $s(t)$. We assume future supply is known. Furthermore, we model supply using a step function defined by a set of rates $S = \{s_0, s_1, \ldots, s_m\}$ and transition points $T = \{t_1, t_2, \ldots, t_m\}$, such that the supply rate transitions from s_{i-1} to s_i at time t_i.

We let $I = \{1, 2, \ldots, n\}$ denote the set of potential consumers, henceforth called *agents*, who arrive over time. Each agent $i \in I$ is interested in a *required amount*, q_i, of the resource and has a *value*, v_i, for this amount. The agent has no additional value for receiving more, and has a value of 0 for getting less than q_i. Additionally, agents can only receive the resource during their *active interval*, $[a_i, d_i]$, where a_i and d_i are the agent's arrival and departure times. Finally, each agent has a *maximum consumption rate* r_i, such that, in a given time interval Δt, agent i can consume at most $r_i \cdot \Delta t$ of the resource. Note that agents are indifferent about when they receive the resource as long as it is within the active interval. Together these form agent i's *type*, $\theta_i = \langle q_i, v_i, r_i, a_i, d_i \rangle$, and we denote the type profile for agents in I by $\theta_I = \{\theta_1, \theta_2, \ldots, \theta_n\}$. This generic model maps nicely into our EV charging scenario, where r_i is the maximum charging speed (in kW) and $[a_i, d_i]$ represents the time interval during which the car is at home and available for charging. Furthermore, this model can be readily applied to a wide range of other domains where expiring resources become available continuously over time, such as the allocation of computational resources in cloud settings, the allocation of bandwidth in networks, or even allocating human resources to projects.

Given this model, we are interested in designing a mechanism for finding a real-time schedule, ρ, where $\rho(i, t) \in \mathbb{R}$ determines the *consumption rate* at which agent i receives the resource at time t. Clearly, such a schedule must satisfy the various constraints outlined above (i.e., total consumption at any time t may not exceed the production rate $s(t)$, agents cannot receive resources outside their active intervals, and they must not exceed their individual maximum consumption rates). Throughout this paper, we are interested in finding a schedule that maximises the overall *social welfare*, i.e., the sum of all agents' values, as this maximises the overall benefit to all agents.

3. SCHEDULING COOPERATIVE AGENTS

In this section, we begin by assuming that agents are completely cooperative and reveal their respective types truthfully to the mechanism on arrival. This allows us to develop a number of scheduling mechanisms which we will later (in Section 4) adapt for settings with strategic agents.

We start by discussing how to solve the *offline* version of the scheduling problem, where the scheduling mechanism has full information about all types in θ_I, including future arrivals. This will form the basis for the more realistic *online* mechanisms we propose later.

3.1 Offline Scheduling

To solve the offline problem, we are simply interested in finding a schedule ρ that maximises the overall social welfare. Formally, we define the social welfare as $W(\rho, \theta_I) = \sum_{i \in I} \delta_i(\rho, \theta_i) \cdot v_i$, where $\delta_i(\rho, \theta_i)$ is an indicator function with $\delta_i(\rho, \theta_i) = 1$ if agent i receives at least its required amount q_i under schedule ρ and $\delta_i(\rho, \theta_i) = 0$ otherwise.

Unfortunately, finding such an optimal schedule ρ is NP-hard.[1] However, we will describe how to find the optimal solution and then discuss a more tractable heuristic procedure. In both cases, we simplify the scheduling problem by partitioning time into non-overlapping intervals within which neither the set of active agents nor the production rate changes. This can be done by using all arrival times, a_i, departure times, d_i, and supply transition times, t_i, to define the partition. Since agents are indifferent about when they

[1]Briefly, this is because it is a generalisation of the NP-hard $1 \mid\mid \sum w_j U_j$ scheduling problem, i.e., minimising the weighted number of tardy jobs [10].

receive the resource within each interval, this allows us to restrict our attention to finding the appropriate consumption rate for each agent within each interval.

3.1.1 *Optimal Offline Scheduling*

We can solve this problem optimally by formulating it as a non-linear programming problem. Here, the objective function is the overall social welfare, the decision variables are the (constant) consumption rates for each agent within each interval, and the constraints are given by the problem domain, as described in Section 2. This problem can be solved using standard optimisation tools — we use ILOG CPLEX in our implementation and denote this mechanism by OPTIMAL. However, due to the inherent complexity of the problem, finding a solution may be time-intensive, and so we propose a faster heuristic in the next section.

3.1.2 *Greedy Offline Scheduling*

To solve the offline scheduling problem more quickly, we first note that it is possible to find a *feasible* schedule for a given set of agents in polynomial time by formulating it as a maximum flow problem (similar to the technique used in [4]) and then using a standard algorithm to solve it. In our implementation, we employ the Edmonds-Karp algorithm and denote this by FINDFEASIBLE (which returns an empty schedule if no feasible schedule exists).

With this, we can apply a greedy approach and iteratively add agents to an overall solution. In more detail, our heuristic greedy algorithm first orders all agents in I in decreasing order of their *value densities*, i.e., v_i/q_i. This order is important, because it is a heuristic indicator of the agent's potential value to the final solution (other indicators such as v_i could be chosen, but we found v_i/q_i to perform particularly well). Then, the algorithm iteratively considers each agent, adding it to a set of selected agents if a feasible schedule can still be found after including that agent (using FINDFEASIBLE). This continues until all agents have been considered and the final feasible schedule including all selected agents is adopted. We refer to this heuristic algorithm as GREEDY.

3.2 Online Scheduling

In an online setting, the scheduling mechanism may only have probabilistic information about future arrivals (e.g., historical data or domain knowledge). The agents' actual types are only revealed to the mechanism at their respective arrival times. Formally, the type profile at time t is given by $\theta_I^{\langle t\rangle} = \{\theta_i \mid i \in I \wedge a_i \leq t\}$. To address this setting, we compare two approaches: model-free scheduling, where no model of future arrivals is available, and model-based scheduling, where the mechanism has access to some statistical information about future arrivals. While this paper focuses on a model-based approach, we use the model-free solutions as a benchmark and to see if having a model of future arrivals provides any additional benefits.

3.2.1 *Model-Free Online Scheduling*

The most straight-forward model-free approach is to simply apply our offline algorithms to $\theta_I^{\langle t\rangle}$. This is done by following the schedule provided by the OPTIMAL or GREEDY offline algorithms, and repeating the algorithm every time a new agent arrives (while updating the required amount of resource for those agents that have been partially satisfied). This allows the algorithm to take advantage of higher-value agents as they arrive, possibly abandoning agents that were previously included in the schedule. Due to the complexity of solving OPTIMAL, we will focus on the heuristic version of this algorithm and denote this by ONLINEGREEDY.

Note that the schedules produced by the offline algorithm, and the FINDFEASIBLE algorithm in particular, may not necessarily result in a high utilisation of the resource in the short-term. This is because *any* feasible schedule is found, which may allocate the resource at any time in the future or at a lower rate than the supply would allow. However, in the online setting, it is desirable to fully utilise the available resources early, so that more agents can be satisfied when they arrive in the future. For this reason, we modify FINDFEASIBLE to use simple heuristic rules that shift all allocations to the earliest possible times, within the problem constraints.

So far, we have concentrated on model-free settings, where the scheduling mechanism does not anticipate future arrivals of agents. However, in many realistic settings, it is reasonable for the mechanism to have some model of the future, for example derived from historical data or the algorithm designer's own knowledge of the system. We explore this in the following section.

3.2.2 *Model-Based Scheduling with Consensus*

Using a model of the future is useful, because it allows the mechanism to adapt its scheduling decisions for likely future events. In particular, it can anticipate the arrival of future high-value agents and accordingly allocate less of the resource to the currently active set of agents when this is likely to be beneficial. Thus, we now assume that our mechanism has some probabilistic knowledge about future arrivals.

As discussed in Section 1, an optimal online scheduling mechanism in these settings is intractable, and so we design a model-based online mechanism based on the Consensus algorithm. This algorithm samples a number of possible future scenarios, solves these using an offline scheduling algorithm, and then uses the majority vote to select which immediate action to take. In addition to being computationally tractable, this approach has the advantage of not requiring a precise model of the future, and so it can be used even in settings where only imprecise statistical information about future arrivals is available. However, adopting Consensus in our setting raises a number of challenges. First, given the continuous-time nature of our setting, it is not clear when to make each online decision. An obvious choice here, and one that we verified experimentally to work well, is to do this every time a new agent arrives in the system.

Second, and more critically, since the resource is continuously divisible, there are an infinite number of possible actions to take. This problem is made even worse if the next action consists of a schedule until the next decision point. To address this problem, instead of voting on schedules, we modify Consensus to vote only on the agents that should be included in the online schedule (and receive their required amount of the resource) and then introduce a second phase in which we produce a feasible schedule with the selected agents. Selecting the set of agents is done iteratively by considering each potential agent in turn, using the greedy heuristic ordering discussed earlier — if the majority of offline solutions include this agent in their schedules, we accept the agent, otherwise we reject it (for the time being). As soon as an agent is accepted, all other candidate agents are tested again, this time enforcing that the accepted agent is part of the solution. This repeats until no more agents are selected, and a final feasible schedule with all accepted agents is calculated and adopted.

This procedure separates the decision of which agents to schedule (as a binary choice for each agent) from the low-level task of finding an overall schedule for all agents. Note that a valid alternative here might be to present all agents to the Consensus scenarios at once, as is done in [8], but this raises several issues. First, the offline algorithms now vote on subsets of agents to include rather than individual agents, which leads to a large decision space that is voted upon. Moreover, a high-value agent may appear in many distinct subsets, but it may not get voted into the solution, due to its votes being divided across these subsets. This issue presents a problem not only for allocation efficiency, but also for incentive compatibility (discussed in Section 4). Serial consideration of the agents avoids these issues.

The full details are given in Algorithm 1. This algorithm keeps in its state k scenarios that are sampled from a suitable model (line 1), where each $\tilde{\theta}_I^j$ is a set of sampled *virtual* agents. For efficiency, we sample these once and re-use them every time Consensus is called. The algorithm also keeps schedule ρ (line 2), which is constructed gradually over time as more agents arrive. Specifically, we assume the function SIGNALARRIVAL($\theta_I^{\langle t \rangle}, t$) is used to notify the algorithm whenever the set of currently known types, $\theta_I^{\langle t \rangle}$, changes. When called, it executes the RUNCONSENSUS function to re-schedule the known agents, but keeps the previous schedule fixed to satisfy the online property ($\rho_{(-\infty,t)}$ is the original schedule ρ, truncated to the time interval $(-\infty, t)$).

RUNCONSENSUS first updates the set of known agent types with the amount of the resource they have received so far following schedule ρ, using an appropriate APPLYSCHEDULE function. All arrival times are also set to t here, to reflect the fact that no past allocations can be altered. Next, the algorithm retains only those virtual agents in the scenarios that arrive in the future. Now, the main loop (lines 11–23) iteratively builds a set of selected agents, C, which is initially empty. To do this, it considers the agents in the same order as our greedy heuristic (to reflect their likely value to the solution), and then solves each of the k scenarios using the offline GREEDY algorithm. If an agent is included in at least half the scenarios, we add it to C. Finally, a feasible schedule for only the agents in C is returned.

Note here that in addition to the set of candidate agents, we add two further parameters to the GREEDY offline algorithm called in line 17. Specifically, C is a subset of the types that must be included in the solution (to reflect the choice of selected agents so far), while the parameter ρ_c will become important in the setting with strategic agents and denotes part of a schedule that has been fixed up to the next decision point. However, here, it is always empty, and, similarly, the set C, holding the set of selected agents, is re-initialised to the empty set at every invocation (line 5). This gives the algorithm full flexibility to change its schedule without having to make any binding pre-commitment choices (a feature that becomes necessary in Section 4).

So far, we have assumed that agents are cooperative and reveal their private types truthfully to the scheduling mechanism. In the next section, we investigate what happens when this is not the case.

4. SCHEDULING STRATEGIC AGENTS

In many realistic settings, agents are self-interested and cannot be assumed to provide truthful information when this is not in their best interest. To examine such settings, we will first introduce some additional terminology from the area of mechanism design (Section 4.1), then we will show why our current Consensus implementation is vulnerable to manipulation (Section 4.2) and finally show how it can be modified to incentivise truthfulness (Section 4.3).

4.1 Model

In this section, we assume that agents can *misreport* their private types to the scheduling mechanism. Specifically, in the EV setting, vehicle owners may plug in their vehicles later than their true arrival times, unplug them earlier, or report needing more charge than is actually the case. We denote by $\hat{\theta}_i = \langle \hat{c}_i, \hat{v}_i, \hat{r}_i, \hat{a}_i, \hat{d}_i \rangle$

Algorithm 1 CONSENSUS Algorithm.

```
 1: $\tilde{\theta}_I^1, \tilde{\theta}_I^2, \ldots, \tilde{\theta}_I^k$                         ▷ Consensus scenarios
 2: $\rho \leftarrow \emptyset$                                                     ▷ Keep track of schedule
 3: $s$                                                                              ▷ Supply
 4: procedure SIGNALARRIVAL($\theta_I^{\langle t \rangle}, t$)
 5:     $\rho' \leftarrow$ RUNCONSENSUS($\theta_I^{\langle t \rangle}, \emptyset, \emptyset, t$)   ▷ Future schedule
 6:     $\rho \leftarrow \rho_{(-\infty,t)} \cup \rho'_{[t,\infty)}$
 7: procedure RUNCONSENSUS($\theta_I^{\langle t \rangle}, C, \rho_c, t$)
 8:     $\theta'_I \leftarrow$ APPLYSCHEDULE($\theta_I^{\langle t \rangle}, \rho, t$)              ▷ Update agents
 9:     for all $j \in \{1, \ldots, k\}$ do
10:         $\tilde{\theta}_I^j \leftarrow \{\theta_i | \theta_i \in \tilde{\theta}_I^j \wedge a_j > t\}$       ▷ Update scenarios
11:     repeat
12:         added $\leftarrow$ false
13:         for all $\theta_i \leftarrow$ GREEDYORDER($\theta'_I \setminus C$) do
14:             if added $=$ false then
15:                 votes $\leftarrow 0$                                                  ▷ Initialise votes
16:                 for all $j \in \{1, \ldots, k\}$ do                                  ▷ Scenarios
17:                     $\rho' \leftarrow$ GREEDY($\tilde{\theta}_I^j \cup \{\theta_i\}, C, \rho_c$)   ▷ Solve
18:                     if $\delta_i(\rho', \theta_i)$ then                                ▷ In?
19:                         votes $\leftarrow$ votes $+ 1$                                 ▷ Vote yes
20:                 if votes $\geq k/2$ then
21:                     added $\leftarrow$ true                                           ▷ Include it
22:                     $C \leftarrow C \cup \{\theta_i\}$
23:     until added $=$ false
24:     return FINDFEASIBLE($C, \rho_c, s$)                                ▷ Return final schedule
```

the type agent i reveals at its *reported arrival time* $\hat{a}_i$, where $\hat{\theta}_i$ may not be equal to θ_i. However, we make a number of reasonable assumptions about the misreports an agent may make, which typically hold in practice. First, it may not report earlier arrivals and it may not report later departures, i.e., it must hold that $\hat{a}_i \geq a_i$ and $\hat{d}_i \leq d_i$. This is reasonable, because the agent must typically be physically present during its active interval to receive the resource, and so its presence can be verified (in the EV setting, a car cannot be plugged in before it physically arrives and, similarly, unplugging earlier than the reported departure could be detected). However, delaying the agent's arrival in the system or departing earlier than necessary are possible manipulations. Second, we assume that agents cannot overstate their maximum consumption rate, i.e., it must hold that $\hat{r}_i \leq r_i$.[2]

Since a scheduling mechanism now uses the reports of all agents as input, denoted by $\hat{\theta}_I$, we write the schedule produced by it as $\pi(\hat{\theta}_I, s)$. Furthermore, to examine and engineer the agents' individual incentives, we also define a *payment function* $x_i(\hat{\theta}_I)$, which determines how much agent i should pay the mechanism on its departure, given the reports of all agents. As is common in mechanism design, we use this payment function to ensure truthfulness. Given this notation, the *utility* of an agent i is $u_i(\pi, x, \theta_i, \hat{\theta}_I, s) = \delta_i(\pi(\hat{\theta}_I, s), \theta_i) \cdot v_i - x_i(\hat{\theta}_I)$, where we note that δ_i here depends on the real type θ_i.

Now, to address potential manipulation by the agents, we are interested in a property called *dominant strategy incentive compatibility* (DSIC). Briefly, if a mechanism π and payment function x are DSIC, this means that it is best for every agent to report their true type, i.e., $\hat{\theta}_i = \theta_i$, regardless of what everyone else reports. Formally, DSIC holds for π and x if and only if it is

[2]This is justified in settings where receiving the resource at a higher rate than r_i either carries a high intrinsic penalty (as is the case in the EV charging setting, where it may damage the expensive battery) or can be detected by the mechanism, for example if not all of the allocated resource is consumed. In these settings, schedules can be constructed in such a way that every agent receives the resource at its maximum rate for some time. However, to simplify the exposition, we do not include such a scheduling mechanism here.

true that $u_i(\pi, x, \theta_i, \hat{\theta}_{-i} \cup \{\theta_i\}, s) \geq u_i(\pi, x, \theta_i, \hat{\theta}_{-i} \cup \{\hat{\theta}_i\}, s)$, $\forall \theta_i, \hat{\theta}_i, \hat{\theta}_{-i}, s$, where we use $\hat{\theta}_{-i}$ to denote the reports of all agents apart from i. Additionally, to ensure participation, we also want π and x to be *individually rational* (IR), which means agents never make a loss when participating. Formally, it must hold that $u_i(\pi, x, \theta_i, \hat{\theta}_{-i} \cup \{\theta_i\}, s) \geq 0, \forall \theta_i, \hat{\theta}_{-i}, s$.

To achieve DSIC and IR, there are a number of results we can draw upon [9]. Specifically, in single-valued domains like ours, the mechanism π must be *monotonic* for DSIC and IR to hold, provided unallocated agents are not paid. To define monotonicity in the context of our work, let $\preceq$ be a partial order over the types with $\theta_i \preceq \theta_j \equiv (a_i \geq a_j) \wedge (d_i \leq d_j) \wedge (r_i \leq r_j) \wedge (q_i \geq q_j) \wedge (v_i \leq v_j)$. Then, we say π is monotonic if and only if $\delta_i(\pi(\{\theta_i\} \cup \theta_{-i}, s), \theta_i) = 1 \Rightarrow \delta_i(\pi(\{\theta'_i\} \cup \theta_{-i}, s), \theta'_i) = 1$, $\forall \theta_i \preceq \theta'_i, \theta_{-i}, s$. This means if an agent θ_i is allocated by an allocation mechanism π, it would remain allocated by π if its type was θ'_i, where $\theta_i \preceq \theta'_i$.

Furthermore, [9] shows that, for a mechanism to satisfy DSIC and IR, the payment for each allocated agent must be equal to its *critical value*. Essentially, this is the lowest value an agent could report to the mechanism and still remain allocated. More formally, the payment of agent i must be $x_i(\{\hat{\theta}_i\} \cup \hat{\theta}_{-i}) = \min v'_i$, such that $\delta_i(\pi(\{\theta'_i\} \cup \theta_{-i}, s), \theta'_i) = 1$, with $\theta'_i = \langle q_i, v'_i, r_i, a_i, d_i \rangle$ ($x_i(\{\hat{\theta}_i\} \cup \hat{\theta}_{-i}) = \infty$ if there is no such v'_i). Unfortunately, we now show that Consensus, as presented in the previous section, is not monotonic.

4.2 Failure of Monotonicity

The key problem with Consensus (and the other algorithms presented in Section 3) is that it does not preserve monotonicity for agents with regard to their departure time. Often, more patient agents are delayed in favour of more constrained agents, but new arrivals may mean they are no longer allocated in the future. As a concrete example, assume there are two agents with types $\theta_1 = \langle 1, 1, 1, 0, 1 \rangle$ and $\theta_2 = \langle 1, 100, 1, 0, 2 \rangle$, and the supply is $s(t) = 1$. For sake of argument, assume that none of the scenarios sampled by Consensus expect more arrivals before time 2, so that, when reporting truthfully, both agents 1 and 2 are included in the set of selected agents, C. However, due to its tighter time constraints, the less valuable but also less patient agent 1 is scheduled first in the interval $[0, 1]$.

Now assume that a new agent arrives at time 1 with $\theta_3 = \langle 1, 200, 1, 1, 2 \rangle$ (which may possibly represent an extremely rare event that was not anticipated by Consensus). Due to its higher value, this is now included in C and scheduled in the interval $[1, 2]$, preventing agent 2 from being allocated. Given this outcome, had agent 2 lied about its patience by misreporting its own type as $\hat{\theta}_2 = \langle 1, 100, 1, 0, 1 \rangle$, it would have been allocated. This breaks monotonicity, as $\hat{\theta}_2 \prec \theta_2$, and it is also clear that it does not satisfy DSIC. In more detail, using critical value payments, the utility for agent 2 when misreporting $\hat{\theta}_2$ is $u_2(\pi, x, \theta_2, \{\theta_1, \hat{\theta}_2, \theta_3\}, s) = 100 - 1 = 99$, while a truthful report would result in being unallocated, with $u_2(\pi, x, \theta_2, \{\theta_1, \theta_2, \theta_3\}, s) = 0$.

To address this issue, we now proceed to introduce a number of modifications to Consensus that achieve monotonicity.

4.3 Truthful Allocation

As discussed in Section 1, one approach to making the allocation policy π monotonic is to apply ironing. This involves identifying cases where an allocation is made, but where the allocated agent's type might have been a misreport that violates monotonicity. In this case, the agent remains unallocated and the resource is discarded. For example, if agent 2 from the example above had reported $\hat{d}_2 = 1$, it would be left unallocated and no resource would be allocated in the interval $[0, 1]$. Clearly, this can be highly inefficient, especially since it means that had agent 2's type really been $\theta_2 = \langle 1, 100, 1, 0, 1 \rangle$, it would still remain unallocated. A second disadvantage of ironing is that testing for monotonicity violations is computationally expensive, as a large set of possible misreports has to be considered.

For this reason, we adopt a different approach. Specifically, we ensure that our Consensus algorithm, π, is monotonic in the first place, avoiding the need for ironing. We achieve this through the following modifications.

4.3.1 Pre-Commitments

First, to address the intrinsic disadvantage for patient agents in the current algorithm, we propose the notion of *pre-commitments*. To this end, we modify Consensus to make a firm commitment to schedule agents if they, at any point, receive a majority vote in the offline scenarios. Once this happens, the agent is pre-committed, which means that it is now guaranteed to receive its required resource (specifically, this is added as an additional constraint to all future scheduling decisions). We achieve this by moving the set of selected agents, C, into the state of the Consensus algorithm, instead of re-initialising it at every invocation of Consensus.

In a sense, while ironing punishes impatient agents, using pre-commitments rewards patient agents instead, because their larger flexibility means they are more likely to be included in the sampled scenarios. However, it is important to note that the commitment decisions still depend on the sampled scenarios, and so even a very patient agent may not receive a pre-commitment until it is highly likely that no better agents will arrive in the future. Furthermore, while a commitment to satisfy the agent is made, the time of when to schedule the agent is initially left open. This allows Consensus to flexibly interrupt or re-schedule the agent as necessary, e.g., when an impatient agent with a high value arrives, as long as the pre-committed agent is still satisfied eventually.

4.3.2 Re-evaluation Points

While pre-commitments ensure monotonicity with regard to departure times, agents may still strategically misreport their arrival times to be allocated (thus breaking monotonicity with regard to arrival times). This is because the decision of Consensus is sensitive to the time it is invoked. Whenever a virtual agent is removed from an offline scenario (line 10 of Algorithm 1), this may change the votes of that particular scenario. As a result, an agent might benefit from misreporting its arrival time to exactly the time where the majority of offline solutions would vote in its favour. To address this, we need to re-run the Consensus decision at every possible time where the solution might change for any unallocated agents. As just discussed, this can happen any time a virtual agent is removed from the scenarios, and, for this reason, we introduce additional *re-evaluation points* whenever this happens. More specifically, we re-run Consensus for every arrival time of a virtual agent across all scenarios.[3]

4.3.3 Partly-Fixed Schedules

However, simply re-evaluating Consensus regularly is not sufficient for monotonicity. Since the schedule ρ' produced by FINDFEASIBLE does not necessarily correspond to the schedules produced

[3]In fact, the set of pre-committed agents cannot change at all re-evaluation points, since more than one vote change is required in most cases. However, for space reasons, we do not discuss this in more detail.

Algorithm 2 CONSENSUS-PC with Pre-Commitments.

```
 1: $C \leftarrow \emptyset$                                              ▷ Pre-committed agents
 2: $t_{\text{next}} \leftarrow \min\{a_i \mid \theta_i \in \theta_I^j, \forall j\}$          ▷ Re-evaluation time
 3: procedure SIGNALARRIVAL($\theta_I^{\langle t \rangle}, t$)
 4:     $\rho \leftarrow$ MODIFIEDCONSENSUS($\theta_I^{\langle t \rangle}, C, \rho_{[t,t_{\text{next}})}, t$)
 5: procedure REEVALUATE($\theta_I^{\langle t \rangle}, t$)                  ▷ Called at time $t = t_{\text{next}}$
 6:     $t_{\text{next}} \leftarrow \min\{a_i \mid \theta_i \in \theta_I^j, \forall j\}$
 7:     $\rho \leftarrow$ MODIFIEDCONSENSUS($\theta_I^{\langle t \rangle}, C, \emptyset, t$)
 8: procedure RUNMODIFIEDCONSENSUS($\theta_I^{\langle t \rangle}, \rho_c, t$)
 9:     repeat
10:         $\rho'_c \leftarrow \rho_c$                                   ▷ Old partly-fixed schedule
11:         $\rho' \leftarrow$ RUNCONSENSUS($\theta_I^{\langle t \rangle}, C, \rho_c, t$)
12:         $\rho_c \leftarrow \rho'_{[t,t_{\text{next}})}$
13:     until $\rho_c = \rho'_c$                                      ▷ No more changes
14:     $\rho \leftarrow \rho_{(-\infty,t)} \cup \rho_c$
15:     return $\rho$
```

by the offline scenarios, following it may cause some of the offline solutions to change even before the re-evaluation point. To address this, we force Consensus to now *partly fix* the schedule over the next interval, $\rho_{[t,t_{\text{next}})}$, where t_{next} is the next re-evaluation point, and then immediately re-run Consensus, this time forcing any scheduling solution to allocate at least as much of the resource to the agents as dictated by $\rho_{[t,t_{\text{next}})}$ (this is the additional input ρ_c to the GREEDY function). This continues until no more changes are made to the set of pre-committed agents. In effect, this gives each offline scenario foresight over what will happen over the next time interval and thereby forces them to make any changes in their votes immediately rather than during the interval. For this reason, $\rho'_{[t,t_{\text{next}})}$ needs to remain fixed over this interval, although any spare resource may still be allocated, e.g., to new agents that arrive during $[t, t_{\text{next}})$. While this restricts the flexibility of the scheduling algorithm slightly, it is possible to add arbitrary new re-evaluation points, which lead to higher flexibility, but also require more evaluations of the Consensus algorithm.

A sketch of our modified Consensus algorithm is given in Algorithm 2, which only shows the main differences to Algorithm 1. Here, an additional function REEVALUATE is called for every re-evaluation point t_{next}. Note, also, when the scenarios contain no agents, this algorithm constitutes a truthful version of the model-free algorithm from Section 3.2.1. We will now show that our new Consensus mechanism (along with critical value payments) is DSIC and IR.

THEOREM 1. *Consensus with pre-commitments, re-evaluation points and partly-fixed schedules is DSIC and IR with critical value payments.*

PROOF. We prove this by using the results from [9] and showing that the modified Consensus is monotonic. To this end, we need to show that if an agent with type θ_i is allocated, it would also be allocated if it had a higher type $\theta'_i \succ \theta_i$. We do this by assuming θ'_i differs from θ_i in exactly one dimension:

Required amount ($q'_i < q_i$): Consider the time t that type θ_i is pre-committed. If θ'_i is not yet pre-committed before t, we will show that it will also be pre-committed at t. In more detail, this is because it is always considered at the same time or earlier than θ_i in the order given by GREEDYORDER (both within Consensus and the Greedy algorithm). Since an active uncommitted agent's presence has no effect on other scheduling decisions up to the point it is pre-committed (due to the serial voting procedure we use, which considers active agents independently of each other), each time feasibility is tested in the offline greedy algorithm, θ'_i will be tested with the same set or a subset of the agents that θ_i was tested with. As θ'_i requires a lower amount of the resource, it will therefore appear in at least as many feasible schedules as θ_i, thus receiving at least the same amount of votes in Consensus.

Value ($v'_i > v_i$): The same argument as above applies.

Rate ($r'_i > r_i$): This follows a similar argument as above. Here, feasibility for θ'_i is tested against the same set of constraints whenever θ_i is tested (as their position given by the greedy order is the same). Thus, when θ_i is pre-committed by being selected by the majority of offline scenarios, so will θ'_i (at the latest), because any schedule that is feasible for θ_i will also be feasible for θ'_i.

Arrival time ($a'_i < a_i$): Consider the time t at which θ_i arrives. If it is not pre-committed immediately, it will be at some future re-evaluation point t'. Since future re-evaluation points are independent of a_i or a'_i, θ'_i would also be pre-committed at the same time t'. On the other hand, if θ_i is pre-committed immediately, we can show that θ'_i would have been pre-committed earlier. To show this, we note that θ_i arrives between two re-evaluation points, α and β, for which the schedule will have been partly fixed. Now, assume θ'_i was present at re-evaluation point α and is still not pre-committed. At the latest, it will now be pre-committed once the schedule over $\rho_{[\alpha,\beta)}$ has been partly-fixed. This is because it competes against exactly the same set of virtual agents as θ_i does at time t (by definition of the re-evaluation points), with the same constraints, and so any time an offline solution contains θ_i at time t, it will also contain θ'_i at time α. A similar argument can be made when θ'_i arrives between α and t, where the same conditions hold.

Departure time ($d'_i > d_i$): This follows a similar argument as for the maximum rate. θ'_i is considered at the same position as θ_i within the greedy solution of each offline scenario. As any feasible schedule for θ_i is feasible for θ'_i, it will be pre-committed at the same time as θ_i at the latest.

We can now show that monotonicity holds also across several dimensions. For this, consider any $\theta_i \preceq \theta'_i$ that vary in x dimensions. Then, we can find a set of intermediate types $\theta_i \preceq \theta_i^1 \preceq \ldots \preceq \theta_i^{x-1} \preceq \theta'_i$, where each type varies in at most one dimension with its predecessor. Using the reasoning above, if θ_i is allocated, so must θ'_i. □

4.4 Practical Considerations

For space reasons, we do not give a detailed algorithm for calculating the critical payments here. However, this is relatively straightforward to implement and can be done using similar techniques as in [5, 8]. To give the reader an intuition, x_i can be calculated by simulating the system without agent i's presence in the interval $[t, \hat{d}_i]$, where t is the time agent i is first pre-committed. At each iteration of Consensus in the simulated system, we then first find the minimum value v'_i agent i would need to report to be included by the majority of offline scenarios (this can be done using a binary search on the greedily ordered virtual agents). Then, we calculate the minimum value v''_i the agent would need to report to be considered before the agent j that is pre-committed during that iteration ($v''_i = 0$ if none is pre-committed). The critical value for this particular iteration of Consensus is then $\max(v'_i, v''_i)$. This is repeated for every iteration and every re-evaluation point and the final payment x_i is then simply the minimum of these critical values.

An attractive feature of calculating the payments in this way is that they can be used also when agents depart earlier than anticipated. Rather than charge them arbitrary penalties, x_i can be calculated for the actual interval the agent was present and for the actual resource received. This allows agents to leave the system when necessary (e.g., in case of an unforeseen emergency), without imposing heavy fines.

5. EXPERIMENTAL EVALUATION

In this section, we apply our mechanism to scheduling the charging of EVs, using data from the first large-scale real-world trial of EVs in the UK. The purpose of this is two-fold. First, we are interested in quantifying the benefit of using a model of future arrivals in these settings. Second, we wish to determine how an incentive-compatible mechanism compares to one that assumes cooperative agents.

5.1 Experimental Setup

To evaluate our mechanisms in a realistic setting, we utilise data gathered during the CABLED (Coventry And Birmingham Low Emissions Demonstration) project.[4] As part of this project, EVs fitted with data loggers were given to the public, in order to study their daily travel and charging patterns. In our experiments, we sample actual recorded journeys from this data set to yield realistic agent arrival and departure times, and we use the actual battery charges consumed during journeys to determine an agent's required amount of electricity. We also use these distributions to (independently) sample scenarios for the Consensus-based mechanisms. Finally, we set all maximum charging rates to 3kW to reflect the capabilities of the EVs used in the trial.

Regarding an agent's value, we distinguish between two types of agents. The first, *low-value* agents, are willing to pay between £0.05 and £0.15 per kWh in their required amount (determined uniformly at random). These would rather remain uncharged by their departure time than pay much more than the usual price of electricity. The second, *high-value* agents, are willing to pay up to £1.50 – £2.50 per mile that they are planning to travel (as given by the sampled CABLED data). These constitute agents that have to travel at their departure time, for example to reach their place of work, and so their valuation represents the approximate cost of having to take a taxi instead. We will vary the relative proportions between the groups to represent varying levels of heterogeneity in the agent population.

Finally, to simulate the production rate $s(t)$, we use the average electricity consumption profile of a small neighbourhood consisting of 50 households, and allow any spare electricity to be used for EV charging whenever consumption drops to below 80% of the peak consumption. This simulates the constraints of the local transformer (with an additional safety margin to account for unexpected fluctuations in consumption), and means that no electricity is available during the peak hours of the early evening, but considerable spare capacity is available during the night.[5]

5.2 Benchmarks

To evaluate our mechanism, we use several benchmarks:

- FAIRCONTENTION: This mechanism divides the available electricity equally between all present uncharged agents. We assume agents unplug as soon as their required amount q_i is reached. As such, this represents a naïve scheduling mechanism that can be easily implemented without requiring agents to report their types.

- ONLINEGREEDY: This *model-free, non-truthful* mechanism uses the heuristic greedy algorithm to schedule the present agents (as described in Section 3.2.1).

- CONSENSUS: This *model-based, non-truthful* mechanism uses Consensus, as given by Algorithm 1. We generate 20 scenarios, as this obtains good results in practice.

- ONLINEGREEDY-PC: This *model-free, truthful* mechanism uses the ONLINEGREEDY algorithm to schedule the present agents, along with pre-commitments to ensure DSIC.

- CONSENSUS-PC: This *model-based, truthful* mechanism uses our modified Consensus algorithm (Algorithm 2).

5.3 Results

The full results of our experiments are shown in Figure 1. Here, we vary the number of EVs within the neighbourhood from 0 to 100 to represent different levels of demand,[6] and we show the results for different proportions of high-value agents, $\nu = 0.0$, $\nu = 0.25$ and $\nu = 0.75$. For statistical significance, we repeat all experiments 1000 times and plot 95% confidence intervals. We chose average social welfare (excluding payments) as the key performance metric, as this is the overall utility generated by each mechanism, and we normalise it to the performance of the offline OPTIMAL.[7] As this uses full information about future arrivals, it serves as a useful upper bound of any mechanism.

First, we consider the performance of FAIRCONTENTION (in green). Clearly, this achieves a very poor performance as soon as demand increases within the neighbourhood. When there are only 25 EVs, its performance drops to around 50% of OPTIMAL, eventually reaching an overall performance of only 5%–10%, depending on the setting. This is because the mechanism considers neither the deadlines nor the charging requirements of the users and so this highlights the need to schedule EVs in a more informed manner.

The simple non-truthful mechanisms, GREEDY and CONSENSUS (both shown in blue), achieve a significantly better performance, because they greedily select a promising set of agents to charge (guided by their value densities) and produce a feasible schedule that considers the deadlines and charging requirements of the agents. By doing this, they consistently achieve around 93% of OPTIMAL. Surprisingly, there is no significant difference between the two mechanisms, implying that the use of a model is not necessary in these settings. This is because the strategies have considerable flexibility in responding to new arrivals in the system by adapting their schedules and so there is little benefit in anticipating these beforehand.

Finally, we consider the two truthful mechanisms, GREEDY-PC and CONSENSUS-PC (both shown in red), which are the main focus on this paper. Here, we note that GREEDY-PC initially performs well, but its performance decreases quickly in settings with high demand. This is particularly pronounced when there are only a few high-value agents, i.e., $\nu = 0.25$, where it achieves less than 35% of the optimal. This is because this strategy does not use a model of the future when making pre-commitments. Thus, in the setting with $\nu = 0.25$, it will pre-commit even to low-value agents, which may then prevent it from accepting high-value agents in the future.

[4]See http://cabled.org.uk

[5]We use real consumption data of domestic households published by SCE for June 2010 (http://www.sce.com/).

[6]Beyond 50, this means that some households own more than one EV. This allows us to evaluate the mechanism in settings with very high demand, and we assume that there is no collusion between the EVs within a household.

[7]As OPTIMAL is too computationally intensive in larger settings, we also applied the offline GREEDY approach. For consistency, the data shown in the graphs uses the GREEDY data throughout as an approximation of the optimal, but we verified experimentally that there is no statistically significant difference in those settings where we ran both algorithms.

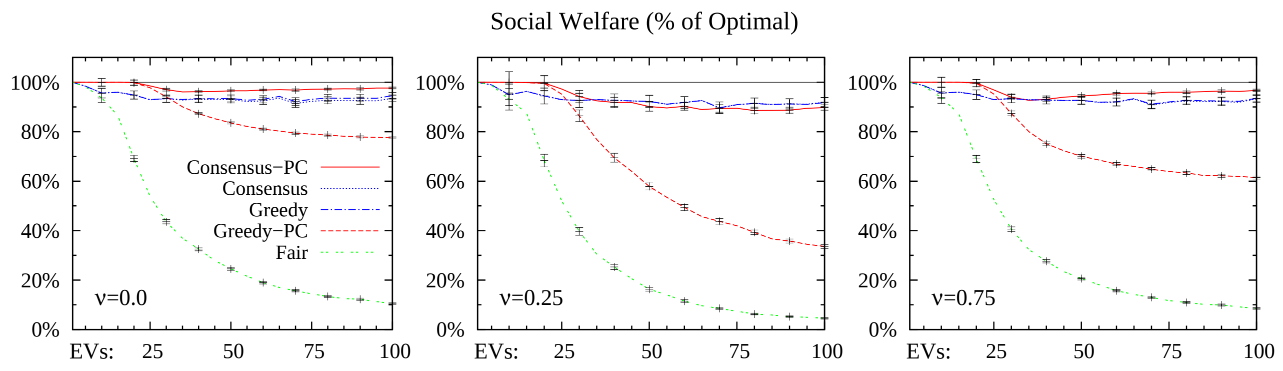


Figure 1: Average social welfare as number of EVs is increased and for various levels of heterogeneity.

In contrast, the model-based truthful mechanism we propose, CONSENSUS-PC, performs significantly better, consistently achieving 92–97% of OPTIMAL. This implies that using a model is critical, in order to make the use of pre-commitments viable in realistic settings. This is because pre-commitments are irrevocable decisions and so the mechanism must be confident that they are unlikely to have a detrimental impact on future scheduling decisions. While we do not consider the runtime performance of our mechanism in detail in this paper, we note that simulating a 24-hour day with 100 cars takes only a few seconds on a standard PC, indicating that it is feasible even in larger settings.

To conclude, we note also that CONSENSUS-PC sometimes outperforms the non-truthful benchmarks, implying that the use of pre-commitment can be beneficial even when DSIC is not a requirement. This is because it forces the algorithm to choose and concentrate on a set of agents, ensuring that these are fully charged. In contrast, the more flexible GREEDY and CONSENSUS re-evaluate their chosen set of agents each time. This means they can initially start charging some vehicles, which are later displaced by others with a higher value density (but not necessarily higher value). Thus, some amount of electricity is wasted, as no value is derived from partially charged vehicles.

6. CONCLUSIONS

In this paper, we considered an online setting where self-interested agents compete for a limited, expiring resource that is continuously being produced. To address shortcomings in existing work, we modified the well-known Consensus algorithm and introduced the novel concept of pre-commitments to ensure incentive compatibility.

Furthermore, we showed how our mechanism can be applied to the challenging problem of scheduling the charging of electric vehicles, a key bottleneck for the widespread adoption of EVs. In experiments using real data, we demonstrated that our mechanism considerably outperforms approaches that divide electricity equally between cars (without considering the owners' individual preferences), and we showed that the cost of ensuring incentive compatibility is small when coupled with a probabilistic model of the future.

However, our work has implications beyond the domain of electric vehicle charging. It can be applied in many application areas with expiring resources, including the scheduling of computational jobs in cloud settings, the allocation of bandwidth in networks or even allocating employees' time to projects within a business.

In future work, we plan to deploy our mechanism in real-world trials, which will focus particularly on the design of appropriate interfaces to allow car owners to express their preferences and interact with the mechanism in a non-obtrusive manner. Other relevant extensions include dealing with multi-value domains, where the utility an agent derives depends on the amount of resource received, and we will explicitly explore other application areas for our mechanism.

7. ACKNOWLEDGEMENT

This work was funded by the ORCHID (`www.orchid.ac.uk`) and iDEaS projects (`www.ideasproject.info`).

8. REFERENCES

[1] R. Bent and P. Van Hentenryck. The value of consensus in online stochastic scheduling. In *ICAPS'04*, 2004.

[2] K. Clement-Nyns, E. Haesen, and J. Driesen. The impact of charging plug-in hybrid electric vehicles on a residential distribution grid. *IEEE Transactions on Power Systems*, 25(1):371–380, 2010.

[3] F. Constantin and D. Parkes. Self-correcting sampling-based dynamic multi-unit auctions. In *EC'09*, pages 89–98, 2009.

[4] A. Federgruen and H. Groenevelt. Preemptive scheduling of uniform machines by network flow techniques. *Management Science*, 32(3):341–349, 1986.

[5] E. Gerding, V. Robu, S. Stein, D. Parkes, A. Rogers, and N. Jennings. Online mechanism design for electric vehicle charging. In *AAMAS'11*, pages 811–818, 2011.

[6] P. V. Hentenryck and R. Bent. *Online Stochastic Combinatorial Optimization*. MIT Press, 2006.

[7] R. A. of Engineering. *Electric Vehicles: Charged with potential*. Royal Academy of Engineering, 2010.

[8] D. Parkes and Q. Duong. An ironing-based approach to adaptive online mechanism design in single-valued domains. In *AAAI'07*, volume 22, page 94, 2007.

[9] D. C. Parkes. *Algorithmic Game Theory*, chapter Online mechanisms, pages 411–439. Cambridge University Press, 2007.

[10] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, 3rd edition, 2008.

[11] R. Porter. Mechanism design for online real-time scheduling. In *EC'04*, pages 61–70, 2004.

[12] S. Vandael, K. D. Craemer, N. Boucké, T. Holvoet, and G. Deconinck. Decentralized coordination of plug-in hybrid vehicles for imbalance reduction. In *AAMAS'11*, pages 803–810, 2011.

Efficient Crowdsourcing Contests

Ruggiero Cavallo
Yahoo! Research
111 West 40th Street
New York, NY 10018
cavallo@yahoo-inc.com

Shaili Jain
Computer Science Department
Yale University
New Haven, CT 06520
shaili.jain@yale.edu

ABSTRACT

A principal seeks production of a good within a limited time-frame with a hard deadline, after which any good procured has no value. There is inherent uncertainty in the production process, which in light of the deadline may warrant simultaneous production of multiple goods by multiple producers despite there being no marginal value for extra goods beyond the *maximum quality* good produced. This motivates a *crowdsourcing* model of procurement. We address efficient execution of such procurement from a social planner's perspective, taking account of and optimally balancing the value to the principal with the costs to producers (modeled as effort expenditure) while, crucially, contending with self-interest on the part of all players. A solution to this problem involves both an algorithmic aspect that determines an optimal effort level for each producer given the principal's value, and also an incentive mechanism that achieves equilibrium implementation of the socially optimal policy despite the principal privately observing his value, producers privately observing their skill levels and effort expenditure, and all acting selfishly to maximize their own individual welfare. In contrast to popular "winner take all" contests, the efficient mechanism we propose involves a payment to every producer that expends non-zero effort in the efficient policy.

Categories and Subject Descriptors

J.4 [**Social and Behavioral Sciences**]: Economics

General Terms

Economics, Algorithms, Human Factors

Keywords

mechanism design, crowdsourcing, contests, social welfare

1. INTRODUCTION

An increasingly common model of procurement has multiple agents simultaneously produce versions of a desired good at the behest of a principal who seeks the highest-quality version. This model's popularity has soared as part of the Internet phenomenon known as *crowdsourcing*. In recent years, the number of websites making and facilitating open calls for solutions to tasks such as logo design, software development, and image labeling has grown tremendously; examples include Amazon Mechanical Turk, Taskcn, Topcoder, 99designs, Innocentive, CrowdCloud, and CrowdFlower, to name a few. These developments have reinvigorated a line of research in the field of microeconomics known as *contest design*. The model of a contest matches the standard approach to crowdsourcing: many agents simultaneously exert effort to submit a solution in competitive pursuit of a reward, where the "winner" is dependent on the relative submission qualities.[1] To date, most previous research has focused on maximizing the principal's utility: the goal is to to procure the best submission for the lowest possible price. In contrast, we here consider the crowdsourcing problem from an efficiency standpoint, adopting the perspective of a *social planner* that seeks to maximize social welfare. And in contrast with the standard "winner take all" contest methods, the efficient scheme we derive involves a payment for every agent that expends non-zero effort in the efficient policy.

To motivate the crowdsourcing paradigm from an efficiency standpoint, assuming rational players,[2] uncertainty and deadlines must play a central role. If these factors were not present then the redundant production inherent to the paradigm would be purely wasteful; one could (and should) alternatively order production sequentially rather than simultaneously, stopping further production when the costs are no longer outweighed by the expected gains given the "quality in hand". So we adopt a model in which the principal seeks production of a good within a single unit of time (corresponding to the span required for production of a single good), after which any goods obtained are of no value. There is inherent uncertainty in production, which may warrant simultaneous production of multiple goods. However, if multiple goods are produced there is no marginal value beyond that of the *maximum quality* good produced. Producers (henceforth, "agents") may have varying *skill* and also make a choice about how much *costly effort* to expend on production: higher effort and skill leads to production of a good with greater expected quality, all else equal.

The principal and all agents are presumed to be self-

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

[1]*All-pay auctions* are also highly related, with the key difference being that each agent's cost there is a payment that generates "revenue" enjoyed by the seller, whereas in a typical contest only the highest-quality-producing agent directly benefits the principal.

[2]While we make this assumption, we do not deny that in practice "irrational" behavioral factors may contribute to the success of many crowdsourcing marketplaces.

interested, which gives rise to a problem of incentives. Achieving efficiency thus has two components: determining an optimal effort policy for the agents, given the value of the principal and the various agent skill levels; and designing a payment mechanism in which no individual can improve his expected utility by misreporting private information (or exerting effort other than what the efficient policy prescribes, in the case of the agents). For a setting where agents have no private information about skill, we derive an efficient, individually rational, and budget-balanced solution—a novel achievement to the best of our knowledge. When skill is private we prove that extending this result is impossible, but we show how a result from the recent mechanism design literature can be applied to yield success if agents can be forced to make commitment decisions prior to learning their skill levels, i.e., if *ex ante* individual rationality suffices.

1.1 Related Work

There has recently been work explicitly addressing the theory of crowdsourcing in a model, like ours, where agents have private skill information and choose an effort level. DiPalantino and Vojnovic [2009] make the connection to all-pay auctions and model a market with multiple contests, considering the principal's optimization problem in the limit-case as the number of agents and contests goes to infinity. Archak and Sundararajan [2009] and Chawla, Hartline, and Sivan [2011] focus on the design of a single contest; the problem consists of determining how many prizes should be awarded, and of what value. Chawla et al. make the connection between crowdsourcing contests and optimal auction design, finding that the optimal crowdsourcing contest is a virtual valuation maximizer.

However, these papers consider a *deterministic* model of quality as a function of effort and skill, under which, if the principal's value is proportional to the *maximum* quality over the produced goods, the crowdsourcing paradigm itself is not well-motivated from an efficiency standpoint. And while in this paper we are concerned with *social* welfare, this prior work is geared towards maximizing utility of the principal alone and is unconcerned with the cost to the agents. This focus is characteristic of the broader literature: in both computer science and economics prior work has, for the most part, focused on maximizing submission quality, whether it be the total sum of submission qualities [Moldovanu and Sela, 2001; 2006; Minor, 2011], the top k submissions less the monetary reward [Archak and Sundararajan, 2009], or only the highest quality submission [Moldovanu and Sela, 2006; Chawla *et al.*, 2011].

This ties in with the extensive literature in economics devoted to the design of optimal *contests*. Many of these works consider a contest model where the prize value is known to all players [Tullock, 1980; Moulin, 1986; Baye *et al.*, 1996], while others adopt a model of incomplete information with respect to the prize [Weber, 1985; Hilman and Riley, 1989; Krishna and Morgan, 1997]. There has also been work on research tournaments that award a single prize. For instance in [Fullerton and McAfee, 1999] agents have a cost of production—drawn from a known distribution—that becomes common knowledge after a first round in which agents simultaneously decide whether to participate; then in a second round agents decide how much effort to exert given the common-knowledge costs.

A related line of work uses contests to extract effort under a hidden action [Lazear and Rosen, 1981; Green and Stokey, 1983; Nalebuff and Stiglitz, 1983]. Similar to our work, the output is a stochastic function of the unobservable effort, but the setting is different in that the principal obtains value from the *cumulative* effort of the agents, rather than just the maximum result.

Finally, this work overlaps with the broader agenda of incentives in peer production systems, where there has been work addressing incentives in question and answer forums, human computation, etc. [Jain and Parkes, 2008; Jain *et al.*, 2009; Ghosh and McAfee, 2011; Ghosh and Hummel, 2011].

1.2 Preliminaries

In the crowdsourcing paradigm multiple units of a good are simultaneously produced and submitted to a principal. There is a set of agents $I = \{1, \ldots, n\}$ capable of producing goods, where each $i \in I$ has private skill level $s_i \in [0, 1]$. Agents can expend variable effort on production of the good. If an agent attempts production, a good is produced with quality that is a priori uncertain but is a function of the agent's skill and effort expended.

Quality is identified with value to the principal in dollar-terms. The probability distribution over *relative* quality, given any skill and effort levels, is publicly known, but the *absolute* quality in terms of value to the principal is not. The principal has private type $v \in \Re^+$, a scale factor corresponding to his value for the maximum quality good that could possibly be produced, and this, given the known distribution over relative qualities, defines the distribution over absolute quality (henceforth just "quality") corresponding to the principal's value.[3] An effort level δ_i is identified with the *dollar value in costs ascribed to it by agent i*. For simplicity we assume that $\delta_i \in [0, 1]$, $\forall i \in I$.[4] Then, given a $v \in \Re^+$, skill level s_i, and effort level $\delta_i \in [0, 1]$, we denote the p.d.f. and c.d.f. over resulting quality as $f^v_{s_i,\delta_i}$ and $F^v_{s_i,\delta_i}$, respectively. We assume symmetry across bidders in the sense that skill is the only differentiating factor; i.e., for two agents with the same skill level applying the same effort, the distribution over quality is the same (though there is no presumed correlation so the resulting quality may differ).

We will make the natural assumption that for an agent with any given skill level, more effort has first-order stochastic dominance over less effort with respect to quality, i.e.:

$$\forall s_i,\ \forall 0 \le \delta_i < \delta'_i \le 1,\ \forall x \in [0, v],\ F^v_{s_i,\delta_i}(x) \ge F^v_{s_i,\delta'_i}(x), \quad (1)$$

and also that, given any effort level, more skill has first-order stochastic dominance over less skill with respect to quality:

$$\forall \delta_i,\ \forall 0 \le s_i < s'_i \le 1,\ \forall x \in [0, v],\ F^v_{s_i,\delta_i}(x) \ge F^v_{s'_i,\delta_i}(x) \quad (2)$$

Because agents are self-interested there is a problem of incentives: v and s_i (for each i) are private information, and expended effort is privately observed. We adopt a quasi-linear utility model and assume all players are risk-neutral. Given our identification of the quality of the good with the dollar value ascribed to it by the principal, and effort level δ_i with the dollar value in costs ascribed to it by agent i, quasilinearity implies that the principal's utility equals the quality of the good procured minus any payments he must make, and each agent's utility equals any payment he receives minus the effort he expends.

[3]E.g., if quality q ranges in $[0, 1]$, absolute quality is vq.

[4]The specific range of effort levels is not conceptually important; it is the relationship between the effort levels and v that is relevant for determining an optimal policy.

We are concerned with the socially optimal choice of effort level for each agent, where in light of our utility model the appropriate optimization is the *maximum* quality level of the goods produced minus total effort expended; i.e., letting $\mathcal{Q}_i(v, s_i, \delta_i)$ be a random variable representing the quality level produced by $i \in I$ who has skill level s_i and expends effort δ_i (with v the principal's value), we seek to maximize:

$$\mathbb{E}[\max_{i\in I} \mathcal{Q}_i(v, s_i, \delta_i)] - \sum_{i\in I} \delta_i \qquad (3)$$

An effort policy is a function of the principal's value and the agents' skill levels. We let $\delta^*(v, s)$ denote an *efficient* policy, i.e., a vector of effort levels that maximizes Eq. (3) given values of v and $s = (s_1, \ldots, s_n)$; when context is clear we will write δ_i^* as shorthand for $\delta_i^*(v, s)$. Given our quasilinear utility model, a policy δ^* that maximizes Eq. (3) maximizes the expected sum of utilities and is Pareto efficient.

At various points we will consider a restricted setting where skill is constant (and publicly known) throughout the population of agents; we call this the *constant skill case.*

2. EFFICIENT EFFORT POLICIES

In this section we address the problem of computing an efficient policy given full knowledge of the principal's value v and agent skill levels s, and given that agents will execute the effort policy that is prescribed. We defer to Section 3 the question of how to implement such a policy in the context of a principal and agents that are self-interested and strategic. We will make heavy use of the following lemma, which demonstrates sufficient conditions under which *extreme-effort* policies—those that involve only total (1) or null (0) effort by each agent—are optimal. In this section we make the technical assumption that the cumulative distribution over quality, evaluated at any particular quality level, is differentiable with respect to effort δ_i.

LEMMA 1. *For arbitrary $i \in I$ with arbitrary skill s_i, for arbitrary skill and effort levels of the other agents and value v for the principal, fixing an arbitrary effort policy for agents other than i, amongst effort levels within arbitrary interval $[a, b] \subseteq [0, 1]$ it is either optimal for i to expend effort $\delta_i = a$ or optimal for i to expend effort $\delta_i = b$ if the following holds: $\forall\beta \in [0, v]$, $\forall\epsilon \in [a, b)$,*

$$-\frac{\partial}{\partial\delta_i}\Big(\int_\beta^v F^v_{s_i,\delta_i}(x)\,dx\Big)\Big|_{\delta_i=\epsilon} \geq 1 \qquad (4)$$

$$\Rightarrow\ -\frac{\partial}{\partial\delta_i}\Big(\int_\beta^v F^v_{s_i,\delta_i}(x)\,dx\Big)\Big|_{\delta_i=k} \geq 1,\ \forall k \in [\epsilon, b] \qquad (5)$$

PROOF. For arbitrary agent $i \in I$, consider arbitrary β representing the maximum quality that is to be realized by the production of the other agents—this is a priori unknown, but a result for *arbitrary* β will demonstrate that regardless of its realization the result holds. Then the expected marginal impact on efficiency from i exerting effort δ_i equals:

$$\int_\beta^v f^v_{s_i,\delta_i}(x)(x-\beta)\,dx - \delta_i \qquad (6)$$

$$= (x-\beta)F^v_{s_i,\delta_i}(x)\Big|_{x=\beta}^{x=v} - \int_\beta^v F^v_{s_i,\delta_i}(x)\,dx - \delta_i \qquad (7)$$

$$= v - \beta - \int_\beta^v F^v_{s_i,\delta_i}(x)\,dx - \delta_i \qquad (8)$$

The first step above is integration by parts. To find the maximum with respect to effort, we consider the derivative with respect to δ_i, i.e.,

$$\frac{\partial}{\partial\delta_i}\Big(v - \beta - \int_\beta^v F^v_{s_i,\delta_i}(x)\,dx - \delta_i\Big) \qquad (9)$$

$$= -\frac{\partial}{\partial\delta_i}\Big(\int_\beta^v F^v_{s_i,\delta_i}(x)\,dx\Big) - 1 \qquad (10)$$

If as δ_i increases this derivative never changes from positive to negative (this is the condition of the Lemma, in Eqs. (4) and (5)), then the maximum lies at one of the extremes, $\delta_i = a$ or $\delta_i = b$, which completes the proof. □

COROLLARY 1. *For environments where the quality distribution functions satisfy the relationship of Eqs. (4–5) in Lemma 1 over the full range of effort levels ($[a, b] = [0, 1]$), an efficient effort policy consists of full-effort participation by a subset of the agents and non-participation by the others.*

Note that the lemma holding for the interval $[0, 1]$ is sufficient *but not necessary* for the optimal policy to involve only extreme-effort (i.e., effort 0 or 1 by all agents). For instance if the condition of the lemma (Eqs. (4) and (5)) does not hold for some β, yet $\forall\delta_i$ the expected quality output for i is less than $\beta - \delta_i$, then the optimal policy would involve non-participation (0 effort) by i when other agents achieve quality β, and so it may still be the case that an optimal policy never involves intermediate effort.

For any constant skill environment (say skill equals $\hat{s}$ for each agent) where we can establish that an extreme-effort policy is optimal, fully determining an efficient policy is easy. We simply need to compute:

$$m^* = \operatorname*{arg\,max}_{m\in\{0,\ldots,n\}} \Big[m\int_0^v F^v_{\hat{s},1}(x)^{m-1} f^v_{\hat{s},1}(x)x\,dx - m\Big] \qquad (11)$$

m^* agents will participate with full-effort and the other $n - m^*$ will not participate (i.e., will apply 0 effort).

When skill is not constant, by Eq. (2) having an agent participate who has *less* skill than one who does not participate could never be optimal. So more generally, for any setting where extreme-effort has been established as efficient we can determine a precise optimal policy by iteratively considering each agent in *decreasing order of skill*, accepting agents for (full-effort) participation until stopping and accepting no more in the ordered list.

In the rest of the section we will show that extreme-effort policies are optimal in important canonical settings, but we first observe that this is not universally the case. Imagine that effort $\delta_i \in [0, 0.05)$ yields quality 0 (with certainty), $\delta_i \in [0.05, 0.3)$ yields quality 0 with probability 0.8 and quality 0.9 with probability 0.2, and $\delta_i \in [0.3, 1]$ yields quality 0.6 with probability 0.8 and quality 0.9 with probability 0.2. An optimal policy for two agents has one agent expend effort 0.3 and the other expend effort 0.05.

2.1 Uniformly distributed quality

We now look at specific distributional settings, starting with one in which quality is uniformly distributed between 0 and the product of the principal's value and the agent's skill and effort. That is, $F^v_{s_i,\delta_i}$ for each $i \in I$ is the uniform distribution over $[0, \delta_i s_i v]$, i.e.,

$$f^v_{s_i,\delta_i}(x) = \begin{cases} \frac{1}{\delta_i s_i v} & \text{if } x \in [0,\ \delta_i s_i v] \\ 0 & \text{otherwise} \end{cases} \qquad (12)$$

We call this the *uniformly distributed quality case.* The range of possible qualities (and the expected quality) increases linearly with skill and effort. We can use Lemma 1 to show that an extreme-effort policy is optimal here.

LEMMA 2. *For the uniformly distributed quality case, for arbitrary skill levels, there is an efficient policy in which each agent $i \in I$ exerts either no effort $\delta_i = 0$ or full effort $\delta_i = 1$.*

PROOF. Consider arbitrary agent $i \in I$ and arbitrary $\beta \in [0, s_i v]$. If we can show that effort $\delta_i = 0$ or $\delta_1 = 1$ for i would yield optimal expected marginal efficiency even if we knew that the other agents would achieve quality β, then the theorem follows. Note that in the case of uniformly distributed quality no effort level less than or equal to $\frac{\beta}{s_i v}$ could possibly be optimal because there would be 0 probability of improving the maximum quality over β (and so if $\beta > s_i v$ then 0 effort by i is clearly optimal). We will now prove the result by using Lemma 1. For arbitrary $\delta_i \in [\frac{\beta}{s_i v}, 1]$:

$$\int_\beta^v F^v_{\delta_i}(x)\,dx = \int_\beta^{\delta_i s_i v} \frac{x}{\delta_i s_i v}\,dx + \int_{\delta_i s_i v}^v 1\,dx \tag{13}$$

$$= \frac{x^2}{2\delta_i s_i v}\Big|_{x=\beta}^{x=\delta_i s_i v} + v - \delta_i s_i v \tag{14}$$

$$= \frac{\delta_i^2 s_i^2 v^2}{2\delta_i s_i v} - \frac{\beta^2}{2\delta_i s_i v} + v - \delta_i s_i v \tag{15}$$

$$= v - \frac{\delta_i s_i v}{2} - \frac{\beta^2}{2\delta_i s_i v} \tag{16}$$

Then:

$$-\frac{\partial}{\partial \delta_i}\Big(\int_\beta^v F^v_{s_i,\delta_i}(x)\,dx\Big) = \frac{s_i v}{2} - \frac{\beta^2}{2 s_i v \delta_i^2}, \tag{17}$$

and this is at least 1 if and only if:

$$\delta_i^2 \geq \frac{\beta^2}{2 s_i v - (s_i v)^2} \tag{18}$$

If this holds for $\delta_i = \epsilon \geq \frac{\beta}{s_i v}$ it holds for $\delta_i = k$ for all $k > \epsilon$. Therefore the distribution satisfies Lemma 1, which tells us that the optimum over the range $[\frac{\beta}{s_i v}, 1]$ occurs at either $\frac{\beta}{s_i v}$ or 1. Since no effort level on the interval $(0, \frac{\beta}{s_i v}]$ could yield positive expected utility and thus be better than effort level 0, the global optimum lies at either $\delta_i = 0$ or $\delta_i = 1$, which completes the proof. □

Then in the constant skill case (with skill normalized to 1) we can compute the optimal number of participants as follows.

THEOREM 1. *For the constant skill, uniformly distributed quality case, a mechanism that elicits maximum-effort participation by m^* arbitrary agents (and 0-effort participation by others) is efficient, where:*

$$m^* = \begin{cases} \lfloor\sqrt{v}\rfloor - 1 & \textit{if } \lfloor\sqrt{v}\rfloor^2 + \lfloor\sqrt{v}\rfloor > v \\ \lfloor\sqrt{v}\rfloor & \textit{otherwise} \end{cases} \tag{19}$$

PROOF. By Lemma 2 an optimal policy will involve full effort by some number $m \in \{0, \ldots, n\}$ agents and 0 effort by the other $n - m$ agents. If m agents participate with full-effort the expected efficiency equals the expected maximum of m draws from $U[0, v]$ minus m, i.e.:

$$\int_0^v \frac{1}{v} mx\Big(\frac{x}{v}\Big)^{m-1} dx - m \ = \ \frac{m}{m+1}v - m \tag{20}$$

Though m is discrete, imagining it as a continuous variable yields a parabola with a single maximum. Taking the derivative with respect to m, we get $\frac{v}{(m+1)^2} - 1$, which has a single positive root at $m = \sqrt{v} - 1$. But since m can only take integer values, when $\sqrt{v}$ is not an integer we have to consider both $\lfloor\sqrt{v} - 1\rfloor$ and $\lceil\sqrt{v} - 1\rceil$ (i.e., $\lfloor\sqrt{v}\rfloor$). The expected value increase of adding a $\lfloor\sqrt{v}\rfloor^{th}$ agent to a group of $\lfloor\sqrt{v} - 1\rfloor$ equals, considering Eq. (20):

$$\Big(\frac{\lfloor\sqrt{v}\rfloor}{\lfloor\sqrt{v}\rfloor + 1}v - \lfloor\sqrt{v}\rfloor\Big) - \Big(\frac{\lfloor\sqrt{v}\rfloor - 1}{\lfloor\sqrt{v}\rfloor}v - (\lfloor\sqrt{v}\rfloor - 1)\Big) \tag{21}$$

$$= \frac{1}{\lfloor\sqrt{v}\rfloor(\lfloor\sqrt{v}\rfloor + 1)}v - 1 \tag{22}$$

This is at least 0 if and only if $\lfloor\sqrt{v}\rfloor^2 + \lfloor\sqrt{v}\rfloor \leq v$, and so the maximum is as characterized by Eq. (19). □

Figure 1 provides a graphic depiction of how the optimal number of agents that should participate relates to the principal's value, for $v \in [0, 100]$.

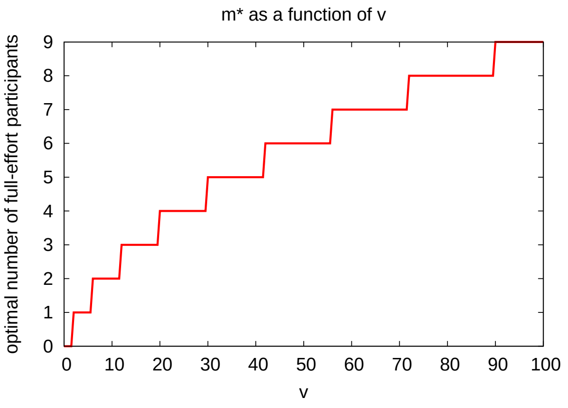


Figure 1: Optimal number of agents to produce a good (with full effort) in the uniformly distributed quality case, as a function of the principal's value v.

2.2 Normally distributed quality

While for more complex distributions beyond the uniform case we would have difficulty demonstrating similar results analytically, we can query whether Lemma 1 holds experimentally. We now consider quality that is normally distributed, over a bounded interval, with mean increasing proportional to effort and skill. Specifically, consider the truncated normal distribution over the interval $[0, v]$, with location parameter μ equal to $\delta_i s_i v$ and scale parameter σ equal to $v/8$; this distribution is illustrated in Figure 2 for various effort levels.

For arbitrary v, β, and δ_i (assuming a fixed $s_i = 1$) we can computationally approximate $\int_\beta^v F^v_{s_i,\delta_i}(x)\,dx$ and accordingly evaluate whether the conditions of Lemma 1 are satisfied. But a more direct approach is equally tractable here: we can simply compute the expected marginal efficiency, given any β, of effort for arbitrarily fine discretizations of the effort space $\delta_i \in [0, 1]$. Then, the next step is to use this approach to check whether extreme-effort is optimal for *all* β in the range $[0, v]$, as this would imply that regardless of the quality obtained by agents other than arbitrary $i \in I$, for i extreme-effort (0 or 1) is optimal. We

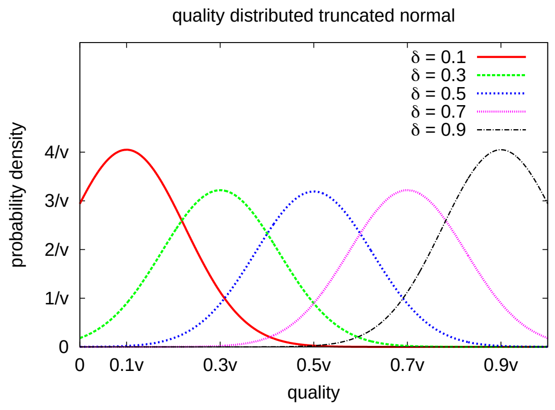


Figure 2: Probability densities over quality for varying degrees of effort, for an agent with skill level 1. Truncated normal with $\mu = \delta_i v$ and $\sigma = v/8$.

checked this by again discretizing the search space; this time the space in question is that of possible (v, β) pairs where β is constrained to fall within $[0, v]$. The results suggest that for all values of v above a very low threshold (2.8), there is no possibility that anything other than extreme effort could be optimal. And we emphasize that with this approach we are only checking certain *sufficient conditions* for optimality of extreme-effort. Finally, given that an extreme-effort policy is optimal, we can easily compute an optimal set of full-effort participants. For the constant skill case this is done according to Eq. (11); see Figure 3 for the results.

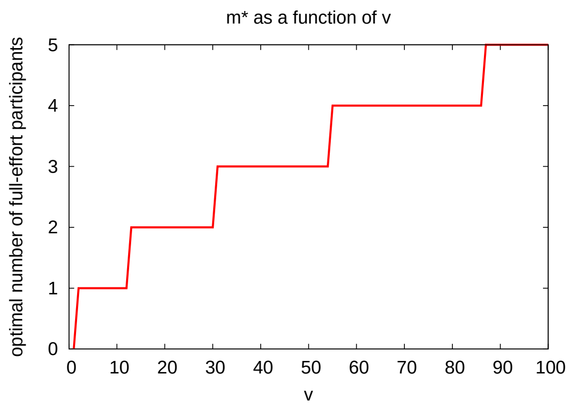


Figure 3: Optimal number of agents that should produce a good (with full effort) as a function of the principal's value v. For the constant skill (equal to 1), truncated ($[0, v]$) normally distributed quality case with $\mu = \delta_i v$ and $\sigma = v/8$.

3. INCENTIVES

We now consider the problem of implementing an efficient policy in a context of selfish players. Since utility in our setting is quasilinear and thus transferable, we can use monetary payments as a tool. Through payments we seek to establish an equilibrium where no agent can gain by doing anything other than what the mechanism asks, as follows:

DEFINITION 1 (INCENTIVE COMPATIBILITY). *A mechanism is incentive compatible if and only if for each player i, given that all other players abide by the mechanism's prescriptions, i's expected utility can never be improved by doing other than what the mechanism prescribes.*

This definition is a generalization of the standard "truthful reporting" definition that is sufficient for mechanisms that only involve sharing of private information. In our setting, one player (the principal) must share private information *truthfully*, while others (the agents) must behave *faithfully* according to what the mechanism prescribes and will also have to share private information truthfully if skill is variable and private knowledge.[5] An incentive compatible mechanism gives us reason to believe that the outcome the mechanism prescribes will occur, given rational agents. But for a mechanism that makes payments there are additional constraints: the mechanism should be weakly *budget-balanced*, in expectation never paying out more than it takes in, since otherwise external subsidies would be required for its implementation. The mechanism should also be *individually rational*, meaning no agent should have negative expected utility in equilibrium from truthfully participating.

3.1 Constant skill

We first consider a context of *constant skill*, where two agents exerting the same effort produce quality according to the same (known) distribution. Recall that an efficient effort policy $\delta^* = (\delta_1^*, \ldots, \delta_n^*)$ is a function of the principal's value and the vector of agents' skill levels s; so in constant skill settings the only "variable" relevant to computation of δ^* is v, and we omit s from all notation. The mechanism we propose is efficient and incentive compatible in such settings, without running a deficit or violating individual rationality. It defines payments that, in some cases, depend on a priori *expected* quality for a given effort level. Recall notation $\mathcal{Q}_i(v, \delta_i)$ for the random variable representing the quality produced when agent i expends effort δ_i, given the principal's value v; $Q_i(v)$ denotes an actual quality level *realized* by i. $Q(v)$ denotes the vector $(Q_1(v), \ldots, Q_n(v))$ and $Q_{-i}(v)$ denotes $(Q_1(v), \ldots, Q_n(v))$ with $Q_i(v)$ excluded; analogously $\mathcal{Q}(v, \delta)$ denotes $(\mathcal{Q}_1(v, \delta_1), \ldots, \mathcal{Q}_n(v, \delta_n))$ and $\mathcal{Q}_{-i}(v, \delta_{-i})$ denotes the same excluding $\mathcal{Q}_i(v, \delta_i)$. For any vector x we let $x^{(k)}$ denote the k^{th} highest element of x.

DEFINITION 2. (CONSTANT SKILL EFFICIENT CROWDSOURCING (CSEC) MECHANISM) *The principal reports v and then efficient effort levels $\delta_1^*, \ldots, \delta_n^*$ are computed. Each agent i is instructed to expend effort δ_i^* on production, and goods are produced with quality levels $Q(v) = (Q_1(v), \ldots, Q_n(v))$. The principal is charged:*

$$\sum_{i \in I} \delta_i^*, \tag{23}$$

agent $h = \arg\max_{i \in I} Q_i(v)$ is paid:

$$\delta_h^* + Q_h(v) - Q^{(2)}(v) - \mathbb{E}[\mathcal{Q}^{(1)}(v, \delta^*) - \mathcal{Q}_{-h}^{(1)}(v, \delta_{-h}^*)], \tag{24}$$

and each other agent $i \in I \setminus \{h\}$ is paid:

$$\delta_i^* - \mathbb{E}[\mathcal{Q}^{(1)}(v, \delta^*) - \mathcal{Q}_{-i}^{(1)}(v, \delta_{-i}^*)] \tag{25}$$

[5]This dual-nature incentive situation appears in many other scenarios; see [Shneidman and Parkes, 2004] and [Cavallo and Parkes, 2008] for precedents in the literature.

The principal pays the sum of the prescribed effort levels; each agent is paid his prescribed effort minus the *expected* difference between the highest quality level overall and the highest quality level achieved by the other agents; each agent is also paid the difference between the *actual* highest quality level produced overall and the highest quality level produced by the other agents—this value is 0 for all agents except he who produces the highest quality good, and thus that agent (h) ends up with a "bonus" $(Q_h(v) - Q^{(2)}(v))$.

Since to compute payments *actual* quality must be known, one can either assume the quality of a produced good given any v is publicly observable or, alternatively, in settings where this is unrealistic the mechanism can be slightly (and harmlessly) modified to have the principal report the quality level of each produced good.

THEOREM 2. *The CSEC mechanism is efficient, incentive compatible, individually rational, and budget-balanced in expectation for constant skill settings.*

PROOF. We start by showing incentive compatibility. The expected utility of the principal, given that he announces $\hat{v}$ and that the agents abide by the mechanism, is:

$$\mathbb{E}[\mathcal{Q}^{(1)}(v, \delta^*(\hat{v}))] - \sum_{i \in I} \delta_i^*(\hat{v}) \tag{26}$$

By efficiency of the computed effort levels (see Eq. (3)), this quantity is maximized with truthful report $\hat{v} = v$.

Now consider arbitrary agent $i \in I$, assume that the principal is truthful and other agents abide by the mechanism and expend effort δ_{-i}^*, and let δ_i denote i's chosen effort level. i is paid the aggregate utility of the other agents minus a quantity completely independent of his behavior; i.e., omitting v from the notation with truthful v understood:

$$\Big[Q^{(1)} - \sum_{j \in I \setminus \{i\}} \delta_j^*\Big] - \tag{27}$$

$$\Big[Q_{-i}^{(1)} - \sum_{j \in I \setminus \{i\}} \delta_j^* - \delta_i^* + \mathbb{E}[\mathcal{Q}^{(1)}(\delta^*) - \mathcal{Q}_{-i}^{(1)}(\delta_{-i}^*)]\Big] \tag{28}$$

Recall that δ_i^* is computed independent of the behavior of i (or any other agent). Therefore i's *expected* utility equals a quantity independent of his control (−Eq. (28)), plus:

$$\mathbb{E}\Big[\mathcal{Q}^{(1)}(\delta_i, \delta_{-i}^*)\Big] - \sum_{j \in I \setminus \{i\}} \delta_j^* - \delta_i \tag{29}$$

By efficiency of δ^*, this is maximized by exerting effort $\delta_i = \delta_i^*$ as prescribed by the mechanism.

Now we consider individual rationality. We now omit δ^* from the $\mathcal{Q}$ notation as well since abiding by the mechanism is understood. The principal's expected utility in the truthful equilibrium equals: $\mathbb{E}[\mathcal{Q}^{(1)}] - \sum_{i \in I} \delta_i^*$, and this is non-negative by efficiency of the policy. Each agent i's expected utility in the truthful equilibrium equals his expected payment minus his expended effort, i.e.:

$$\mathbb{E}\Big[\mathcal{Q}^{(1)} - \sum_{j \in I \setminus \{i\}} \delta_j^*\Big] - \tag{30}$$

$$\mathbb{E}\Big[\mathcal{Q}_{-i}^{(1)} - \sum_{j \in I \setminus \{i\}} \delta_j^* - \delta_i^* + \mathbb{E}[\mathcal{Q}^{(1)} - \mathcal{Q}_{-i}^{(1)}]\Big] - \delta_i^* \tag{31}$$

$$= \mathbb{E}\Big[\mathcal{Q}^{(1)} - \mathcal{Q}_{-i}^{(1)}\Big] - \mathbb{E}\Big[\mathcal{Q}^{(1)} - \mathcal{Q}_{-i}^{(1)}\Big] = 0 \tag{32}$$

Finally, consider the expected aggregate payments received by the social planner. Noting that in the truthful equilibrium Eq. (27) minus Eq. (28) reduces to $\delta_i^* + Q^{(1)} - Q_{-i}^{(1)} - \mathbb{E}[\mathcal{Q}^{(1)} - \mathcal{Q}_{-i}^{(1)}]$, incorporating payments received from the principal in expectation this equals:

$$\sum_{i \in I} \delta_i^* - \sum_{i \in I} \Big(\delta_i^* + \mathbb{E}[\mathcal{Q}^{(1)} - \mathcal{Q}_{-i}^{(1)}] - \mathbb{E}[\mathcal{Q}^{(1)} - \mathcal{Q}_{-i}^{(1)}]\Big) = 0 \tag{33}$$

And so the budget is exactly balanced in expectation. □

Let us consider an example. Imagine there are three agents (with constant skill equal to 1), a principal with value $v = 8$, and uniformly distributed quality. We can use Theorem 1 to determine an optimal policy: since $\lfloor\sqrt{v}\rfloor^2 + \lfloor\sqrt{v}\rfloor = 6 < v = 8$, the optimal policy calls for $\lfloor\sqrt{v}\rfloor = 2$ agents—say agents 1 and 2—to expend effort $\delta_1 = \delta_2 = 1$ and the third to expend effort $\delta_3 = 0$. Imagine that the realized quality levels turn out to be $Q_1 = 3$ and $Q_2 = 5$. The mechanism requires that the principal pay: $\delta_1 + \delta_2 + \delta_3 = 2$. Noting that $\mathbb{E}[\mathcal{Q}^{(1)}(8, (1, 1, 0))] = 16/3$ and $\mathbb{E}[\mathcal{Q}^{(1)}(8, (1, 0))] = 4$, agent 1 is paid: $1 - (16/3 - 4) = -1/3$, i.e., he is charged $1/3$. Agent 2, the maximum quality-producing agent, is paid: $1 + (5 - 3) - (16/3 - 4) = 5/3$. Finally, agent 3 is paid: $0 - (0 - 0) = 0$. Each agent's utility equals his payment minus effort ($-4/3$ for agent 1, $2/3$ for agent 2, and 0 for agent 3); the principal's utility equals $5 - (1 + 1) = 3$; and revenue to the mechanism designer equals $2 + 1/3 - 5/3 = 2/3$. No agent could have gained in expectation from deviating from the mechanism's prescriptions, and although agent 1 was worse off for having participated, in expectation he was not so participation is rational given risk-neutrality.

Perhaps it could be considered a flaw of the mechanism that agents do not have *strict* incentive to participate: their expected utility from doing so is 0. First of all we note that the effort cost δ_i for an agent can be understood to incorporate *opportunity costs*, and can thus be construed as the difference in cost between the given effort level and the value of the agent's "outside option" (which will equal 0 if the agent has no other options).

But we can go further. Note that the principal *does* obtain positive surplus from the mechanism; in fact he is the only player (including the social planner) that does so in expectation. We can seek to distribute this more broadly. Let $\underline{v}$ be the minimum value the principal could *possibly* have, i.e., the greatest value that—independent of the principal's announcement—the mechanism designer *knows* is no greater than the true v (in the worst case $\underline{v} = 0$, but it may be greater). Let:

$$G = \mathbb{E}[\mathcal{Q}^{(1)}(\underline{v}, \delta^*(\underline{v}))] - \sum_{i \in I} \delta_i^*(\underline{v}) \tag{34}$$

We can amend the CSEC mechanism by charging the principal G and paying each agent G/n. Since G is completely independent of the principal's report, charging him thus will not change his incentives. Because quality is monotonically increasing in his value, G is a lower bound (guarantee) on the expected surplus the principal obtains in equilibrium under the CSEC mechanism, and so individual rationality will still hold in the amended mechanism. For any v in the space of possible values, the principal's expected utility will equal:

$$\mathbb{E}[\mathcal{Q}^{(1)}(v, \delta^*(v))] - \sum_{i \in I} \delta_i^*(v) - G \tag{35}$$

$$\geq \mathbb{E}[\mathcal{Q}^{(1)}(v, \delta^*(\underline{v}))] - \sum_{i \in I} \delta_i^*(\underline{v}) - G \tag{36}$$

$$\geq \mathbb{E}[\mathcal{Q}^{(1)}(\underline{v}, \delta^*(\underline{v}))] - \sum_{i \in I} \delta_i^*(\underline{v}) - G = 0, \tag{37}$$

where the first inequality holds by efficiency of δ^*. In expectation the mechanism remains perfectly budget-balanced, while now each agent may obtain positive utility and so will the principal (assuming $v \neq \underline{v}$).

In using this approach we are essentially adopting the technique of [Cavallo, 2006] in which revenue is "redistributed" to the agents in an effort to maintain wealth within the group rather than in the hands of the mechanism designer, without distorting incentives. Here we are seeking to redistribute surplus to the agents from the principal, but the technique is identical to that of [Cavallo, 2006] except instead of redistributing revenue we redistribute surplus.

3.2 Privately known skill

In the more general case where the principal has private value information, the agents have privately observed effort, *and* the agents have private skill information, the incentives problem is significantly more challenging. In fact, we can use the Myerson-Satterthwaite impossibility theorem [Myerson and Satterthwaite, 1983] to demonstrate the impossibility of achieving an efficient, incentive compatible, individually rational, and budget-balanced mechanism.

THEOREM 3. *There exists no mechanism that—for unrestricted quality distributions, private value for the principal, and private agent skill levels—is efficient, incentive compatible, individually rational, and budget-balanced.*

PROOF. We can prove the result via a "reduction" to efficient crowdsourcing from bilateral trade, for which we know by [Myerson and Satterthwaite, 1983] that there is no efficient, incentive compatible, individually rational, and budget-balanced mechanism. Assume for contradiction that the theorem fails. Then for any bilateral trade setting where the seller has value $\theta_s \in [0, 1]$ and the buyer has value $\theta_b \in [0, 1]$ consider the following crowdsourcing problem: the principal has value $v = \theta_b$, there is a single agent i who has skill $s_i = 1 - \theta_s$, and quality is (deterministically) distributed as follows:

$$Q_i(v, s_i, \delta_i) = \begin{cases} v & \text{if } \delta_i \geq 1 - s_i \\ 0 & \text{otherwise} \end{cases} \tag{38}$$

Assume that if the social planner chooses a policy in which quality v would be realized with certainty (given announced skill), the agent can be *compelled* to exert effort until quality v results (despite the planner not being able to observe the actual effort level expended). This only makes solving the crowdsourcing problem *easier*, and so a solution to the full crowdsourcing problem implies a solution to this variant.

Note that in any efficient policy i exerts effort either 0 or $1 - s_i = 1 - (1 - \theta_s) = \theta_s$. If the policy calls for production then the principal gains utility θ_b and the agent loses utility θ_s. Therefore in the efficient policy production occurs if and only if $\theta_b \geq \theta_s$. Moreover note that the strategic situations of the principal and agent in this crowdsourcing problem are *identical* to that of the buyer and seller in the bilateral trade problem, in the sense that the expected utility of each, given any strategy they play, for any strategy played by the other, is identical in either problem. So a payment scheme that is effective for the crowdsourcing problem would constitute a solution for the bilateral trade problem, and this would contradict the Myerson-Satterthwaite theorem. □

This negative result notwithstanding, there is a way forward with a weaker individual rationality concept. Though skill is private, it is not implausible to imagine that agents only learn their skill levels *after* the nature of the project is announced, in which case *ex ante* individual rationality would be sufficient to achieve participation if we force agents to make participation decisions before announcing the nature of the task. And [Cavallo, 2011] provides an efficient mechanism—which here we will call the *ex-ante-commitment mechanism*—that is incentive compatible for arbitrary private values settings and achieves individual rationality ex post of type realization for one player, while achieving individual rationality (and budget-balance) *ex ante* of type realizations for all others. This fits our setting perfectly, since the principal will know his value from the outset, but commitment by the agents may potentially be arranged to occur prior to realization of their types. Our setting has private *action* (effort) as well as private information, but the incentives provided by the mechanism extend.

The ex-ante-commitment mechanism makes payments based on expectations with respect to a prior distribution over agent types, assumed to be shared (a priori) by the mechanism designer and all agents. We use notation $\mathbb{E}_{\tilde{s}}[\cdot]$ to denote the expected value of a quantity with respect to the prior distribution over agents' skill levels, with $\tilde{s}$ a random variable representing the skill vector. Again letting $\underline{v}$ denote the lowest value in the principal's value space, let:

$$G(s) = \mathbb{E}[\mathcal{Q}^{(1)}(\underline{v}, s, \delta^*(\underline{v}, s))] - \sum_{i \in I} \delta_i^*(\underline{v}, s) \tag{39}$$

A derivative of the ex-ante-commitment mechanism for our setting takes the following form:

DEFINITION 3. (PRIVATE SKILL EFFICIENT CROWDSOURCING (PSEC) MECHANISM) *The principal reports v, each agent $i \in I$ reports s_i, and then the efficient effort levels $\delta_1^*, \ldots, \delta_n^*$ are computed. Each $i \in I$ is instructed to expend effort δ_i^*, and goods are produced with quality levels $Q(v) = (Q_1(v), \ldots, Q_n(v))$. The principal is charged:*

$$\sum_{i \in I} \delta_i^* + G(s), \tag{40}$$

and each agent $i \in I$ is paid:

$$\mathcal{Q}^{(1)}(v) - \sum_{j \in I \setminus \{i\}} \delta_j^* - \tag{41}$$

$$\mathbb{E}_{\tilde{s}}\big[\mathcal{Q}^{(1)}(v, \tilde{s}, \delta^*(v, \tilde{s})) - \sum_{j \in I} \delta_j^*(v, \tilde{s}) - \frac{1}{n} G(\tilde{s})\big] \tag{42}$$

The following theorem is essentially a consequence of the main theorem of [Cavallo, 2011], although in that work the strategic element is solely related to private information, while for us there is also a hidden action (effort). But the basic logic extends: Groves mechanisms achieve straightforward *behavior*, which typically consists of truthful reporting but can also encompass other actions, e.g., production effort.

THEOREM 4. *The PSEC mechanism is incentive compatible, individually rational for the principal, individually rational for each agent ex ante of skill level realizations, and budget-balanced in expectation ex ante of skill realizations.*

We demonstrate the workings of the mechanism on the following uniformly distributed quality example: there are two agents, where the prior distribution over each's skill level assigns probability 0.5 to skill 0 and probability 0.5 to skill 1; assume the realized skills are $s_1 = 0$ and $s_2 = 1$. Assume the principal's value space is $[5, 50]$ (so $\underline{v} = 5$), and that the principal's actual value is $v = 10$. In the optimal policy agent 2 participates with full effort and agent 1 does not participate. $G(s) = 2.5 - 1 = 1.5$ and $\mathbb{E}_{\tilde{s}}\big[\mathcal{Q}^{(1)}(v, \tilde{s}, \delta^*(v, \tilde{s})) - \sum_{i \in I} \delta^*_i(v, \tilde{s}) - \frac{1}{n}G(\tilde{s})\big] \approx 2.6$. Imagine that quality 6 is realized. Then the principal pays $1 + 1.5 = 2.5$, obtaining a net utility of 3.5. Agent 1 is paid $6 - 1 - 2.6 = 2.4$, for a net utility of 2.4. Agent 2 is paid $6 - 0 - 2.6 = 3.4$, for a net utility of 2.4. Revenue equals $2.5 - 2.4 - 3.4 = -3.3$.

While the social planner or some agent may end up worse off ex post, as in the example, in expectation given the distribution over types each will gain from participation.

4. CONCLUSION

While most prior work seeks to maximize the utility of the principal alone, in this paper we pursue a crowdsourcing scheme that is *socially* optimal, maximizing the aggregate efficiency to all stakeholders in the system; we believe this holds the potential to bring significant added value to crowdsourcing marketplaces. Our findings and proposals may be of interest from both a theoretical and practical standpoint. From a theoretical perspective, we provide an efficient, individually rational, budget-balanced mechanism in the constant skill case; while we show that this is not possible in the general case, there we describe a mechanism that is efficient, budget-balanced, IR for the principal, and *ex-ante* IR for the producers. The results inform a designer of a crowdsourcing contest how to compute the optimal number of participants, given the principal's value and the agents' distribution over quality, and also tell the designer how to award the payments or prizes. An interesting facet of the mechanism we propose is that if the optimal number of participants is k, then the mechanism should award k payments (or more in the private skill case). This is in contrast with the winner-take-all schemes currently prevalent in crowdsourcing, where the participant who submits the highest quality good is the sole recipient of a lump sum prize.

5. ACKNOWLEDGMENTS

We thank John Langford for a helpful discussion about computing efficient policies.

6. REFERENCES

[Archak and Sundararajan, 2009] Nikolay Archak and Arun Sundararajan. Optimal design of crowdsourcing contests. In *Proceedings of the Thirtieth International Conference on Information Systems (ICIS-09)*, 2009.

[Baye *et al.*, 1996] Michael R. Baye, Dan Kovenock, and Casper G. de Vries. The all-pay auction with complete information. *Economic Theory*, 8:291–305, 1996.

[Cavallo and Parkes, 2008] Ruggiero Cavallo and David C. Parkes. Efficient metadeliberation auctions. In *Proceedings of the 26th Annual Conference on Artificial Intelligence (AAAI-08)*, pages 50–56, 2008.

[Cavallo, 2006] Ruggiero Cavallo. Optimal decision-making with minimal waste: Strategyproof redistribution of VCG payments. In *Proceedings of the 5th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-06)*, pages 882–889, 2006.

[Cavallo, 2011] Ruggiero Cavallo. Efficient allocation through delayed information revelation. working paper, 2011.

[Chawla *et al.*, 2011] Shuchi Chawla, Jason Hartline, and Balusubramanian Sivan. Optimal crowdsourcing contests. In *Proceedings of the 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA-12)*, pages 856–868, 2011.

[DiPalantino and Vojnovic, 2009] Dominic DiPalantino and Milan Vojnovic. Crowdsourcing and all-pay auctions. In *Proceedings of the 10th ACM Conference on Electronic Commerce (EC-09), Stanford, CA*, pages 119–128, 2009.

[Fullerton and McAfee, 1999] Richard Fullerton and R. Preston McAfee. Auctioning entry into tournaments. *Journal of Political Economy*, 107:573 – 605, 1999.

[Ghosh and Hummel, 2011] Arpita Ghosh and Patrick Hummel. A game-theoretic analysis of rank order mechanisms for user generated content. In *Proceedings of the 12th ACM Conference on Electronic Commerce (EC-11), San Jose, CA, USA*, pages 189–198, 2011.

[Ghosh and McAfee, 2011] Arpita Ghosh and Preston McAfee. Incentivizing high quality user generated content. In *Proc. of the 20th International Conference on World Wide Web (WWW-11), Hyderabad, India*, pages 137–146, 2011.

[Green and Stokey, 1983] J. Green and N. Stokey. A comparison of tournaments and contracts. *Journal of Political Economy*, 91(3):349–364, 1983.

[Hilman and Riley, 1989] A. Hilman and J. Riley. Politically contestable rents and transfers. *Economics and Politics*, 1:17–39, 1989.

[Jain and Parkes, 2008] Shaili Jain and David C. Parkes. A game-theoretic analysis of games with a purpose. In *Proceedings of the 4th International Workshop on Internet and Network Economics (WINE-08)*, pages 342–350, 2008.

[Jain *et al.*, 2009] Shaili Jain, Yiling Chen, and David C. Parkes. Designing incentives for online question and answer forums. In *Proc. of the 10th ACM Conference on Electronic Commerce (EC-09), Stanford, CA*, pages 129–138, 2009.

[Krishna and Morgan, 1997] V. Krishna and J. Morgan. An analysis of the war of attrition and the all-pay auction. *Journal of Economic Theory*, 72:343 – 362, 1997.

[Lazear and Rosen, 1981] E. Lazear and S. Rosen. Rank order tournaments as optimum labor contracts. *Journal of Political Economy*, 89:841–864, 1981.

[Minor, 2011] Dylan Minor. Increasing efforts through incentives in contests. Manuscript, 2011.

[Moldovanu and Sela, 2001] Benny Moldovanu and Aner Sela. The optimal allocation of prizes in contests. *American Economic Review*, 91:542–558, June 2001.

[Moldovanu and Sela, 2006] Benny Moldovanu and Aner Sela. Contest architecture. *Journal of Economic Theory*, 126(1):70–96, January 2006.

[Moulin, 1986] Herve Moulin. *Game Theory for the Social Sciences*. New York University Press, 1986.

[Myerson and Satterthwaite, 1983] Roger Myerson and Mark A Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 28:265–281, 1983.

[Nalebuff and Stiglitz, 1983] B. Nalebuff and J. Stiglitz. Prizes and incentives: Towards a general theory of compensation and competition. *Bell Journal of Economics*, 14:21–43, 1983.

[Shneidman and Parkes, 2004] Jeffrey Shneidman and David C. Parkes. Specification faithfulness in networks with rational nodes. In *Proceedings of the 23rd ACM Symposium on Principles of Distributed Computing*, pages 88–97, 2004.

[Tullock, 1980] Gordon Tullock. Efficient rent-seeking. In James M. Buchanan, Robert D. Tollison, and Gordon Tullock, editors, *Towards a Theory of the Rent-Seeking Society*, pages 97–112. Texas A&M University Press, 1980.

[Weber, 1985] Robert Weber. Auctions and competitive bidding. In H.P. Young, editor, *Fair Allocation*, pages 143–170. American Mathematical Society, 1985.

Session 4D
Economies & Markets II

Identifying Influential Agents for Advertising in Multi-agent Markets

Mahsa Maghami
Department of EECS
University of Central Florida
Orlando, FL, USA
mmaghami@cs.ucf.edu

Gita Sukthankar
Department of EECS
University of Central Florida
Orlando, FL, USA
gitars@eecs.ucf.edu

ABSTRACT

The question of how to influence people in a large social system is a perennial problem in marketing, politics, and publishing. It differs from more personal inter-agent interactions that occur in negotiation and argumentation since network structure and group membership often pay a more significant role than the content of what is being said, making the messenger more important than the message. In this paper, we propose a new method for propagating information through a social system and demonstrate how it can be used to develop a product advertisement strategy in a simulated market. We consider the desire of agents toward purchasing an item as a random variable and solve the influence maximization problem in steady state using an optimization method to assign the advertisement of available products to appropriate messenger agents. Our market simulation accounts for the 1) effects of group membership on agent attitudes 2) has a network structure that is similar to realistic human systems 3) models inter-product preference correlations that can be learned from market data. The results show that our method is significantly better than network analysis methods based on centrality measures.

Categories and Subject Descriptors

I.2.11 [**Distributed artificial intelligence**]: Multi-agent systems

General Terms

Algorithms

Keywords

Marketing, Optimization, Multi-agent social simulations

1. INTRODUCTION

The gift of persuasion is a powerful and highly-sought after skill, as evidenced by the fact that individual self-help books in this area, the most famous being *How to Win Friends and Influence People* published in 1936, remain popular. The rise of social media outlets and click-through advertisement opened the door for relatively small groups to influence large numbers of people. Combined with modern data analysis techniques, it is possible to create a detailed social simulation of the population of interest, but the problem of whom to influence remains as an open research question. Particularly in advertisement, indiscriminate mass marketing techniques can lead to negative information cascades about product quality, even if cost efficiency is not an issue. This problem can be framed as a network influence propagation problem; previous work in this area has looked at diverse domains such as information propagation in the Flickr social network [7] and identifying important blogs for marketing [3].

In this paper we present a mathematical analysis of how influence propagation occurs over time and propose a new optimization technique for identifying effective messenger agents in the network that outperforms other network analysis methods while accounting for realistic factors such as group membership and product preference correlation. Following the work of Hung et al. [12, 13], optimization is used along with an analysis of the expected long-term system behavior to assign the advertisement of the available products to appropriate agents in the network. In contrast with previous work on identifying influential nodes for marketing purposes (e.g., [11] and [4]), in this work we model the effects of realistic social factors such as group membership on product adoption. In the analysis presented in [12, 13] for counterinsurgency messaging tactics, there exists a single random variable representing the attitude of agents toward counterinsurgency, but in our work, we use a vector of random variables which represents the desire of each agent toward any single product. This consideration combined with product demand correlations in the market make the analysis and optimization more complicated, but ultimately our approach has the promise of being applicable to a wider variety of social systems.

The paper is organized as follows. Section 2 describes the agent-based model and how it can be used for influencing propagation. Section 3 presents our network generation model that is designed to account for social influence factors present in real human societies such as homophily and group membership. The subsequent section presents our analysis of the system dynamics, and our proposed optimization technique is described in Section 5. We evaluate our method vs. a set of centrality based network analysis techniques in Section 6. We end the paper with an overview of related work in this area and a discussion of future work.

2. MARKET MODEL

To explore the efficiency of the proposed marketing method, we have extended a multi-agent system model, inspired by [12] and [13], to simulate a social system of potential customers. In this model, there is a population of N agents, represented by the set $A = \{a_1, \ldots, a_N\}$, that consists of two types of agents ($A = A_R \cup A_P$). The first type of agent, defined as: $A_R = \{a_r \mid a_r \textit{ is Mutable and } 1 \leq r \leq R\}$, are the *Regular* agents, who are the potential customers. These agents have a changing atti-

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

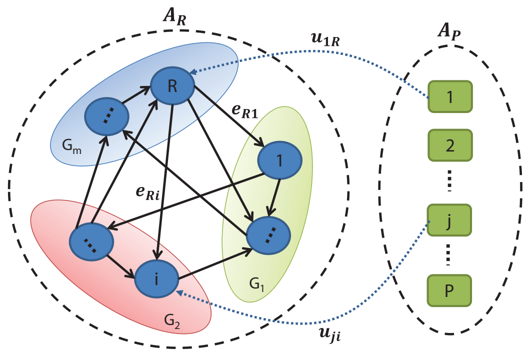


Figure 1: The model of the social system. There exist two types of agents, *Regular* agents (A_R) and *Product* agents (A_P). A static network exists among *Regular* agents, and our problem is to find effective connections between the *Product* (sellers) and *Regular* agents (customers) in order to influence the customers to buy products. *Regular* agents also can belong to different groups in their society (G_m), which modifies the local influence propagation properties.

tude on purchasing products and can be influenced by the *Product* agents who represent salespeople offering one specific product. These agents have an immutable attitude toward a specific product and are defined as: $A_P = \{a_p \mid a_i \textit{ is Immutable and } 1 \leq p \leq P\}$. Figure 1 provides an illustration of the market model.

Each *Regular* agent can be considered as a unique node in the social network, connected by directed weighted links based on the underlying interactions with other agents. The connection between the *Regular* agents is modeled by an adjacency matrix, $\mathbf{E}$, where $e_{ij} = 1$ is the weight of a directed edge from agent a_i to agent a_j. The in-node and out-node degrees of agent a_i are the sum of all in-node and out-node weights, respectively ($d^i_{in} = \sum_{a_j \subseteq A_R} e_{ji}$ and $d^i_{out} = \sum_{a_j \subseteq A_R} e_{ij}$). This network is assumed to follow a power law degree distribution like many human networks, and is generated synthetically as explained in Section 3.

We model the desire of an agent, a_i, to buy an item or consume a specific product, p, as a random variable denoted by $x_{ip} \in [-1\ 1]$. As there exist P items in the environment, each agent is assigned a vector of random variables, $\overrightarrow{X_i}$, representing the attitude or desire of the agent toward all of the products in the market.

Within the social network there are different groups of *Regular* agents; these groups could represent demographic groups or other types of subcultures. Agents from the same group are more effective at influencing each other. To model this, the social system contains m different long-lasting groups, $G_1, \ldots, G_m$, and each agent i is designated with a group membership, G_i.

Here, we do not attempt to capture a rich social-cultural behavior model of these interactions, but rather view the model simply as a function $\mathcal{F} : \mathrm{G_i} \longrightarrow S_i$, mapping the group label of agents, $\mathrm{G_i}$, to a social impression, S_i, that affects link formation and influence propagation, which we designate as the group value judgment. This value represents the agents' judgments on other groups and is based on observable group label of the agent rather than real characteristics of the person. We assume that the impression of different groups has been learnt by agents beforehand therefore each agent has a unique vector of judgment values, noted as $\overrightarrow{S_i} = S_1, S_2, \ldots, S_m$, to indicate the judgment of each agent on different groups in the simulated society.

Moreover, in real life there is a correlation between the user demand of different products in the market. The desire of customers for a specific product is related to his/her desire toward other similar products. To model this correlation and consider its effect in our formulation, we designate a matrix $\mathbf{M}$ that identifies the relationship between demands among advertised items and can be shown as:

$$\mathbf{M} = \begin{pmatrix} m_{11} & \ldots & m_{1P} \\ \vdots & \ddots & \vdots \\ m_{P1} & \ldots & m_{PP} \end{pmatrix}$$

where m_{ij} indicates the probability of having desire toward item j assuming the agent already has a desire for item i. We assume that this matrix is known beforehand and has been modeled by the advertisement companies by tracking the users and applying user modeling.

In the market, the companies are trying to select a set of connections between the A_P agents and A_R agents, in such a way to maximize the long term desire of the agents for the products. We define a simple decision variable u_{ji}, where

$$u_{ji} = \begin{cases} 1 & \textit{Product } j \text{ connects to } \textit{Regular} \text{ agent } i, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Note that the links between *Product* agents and *Regular* agents are directed links from products to agents and not in the opposite direction, and that *Product* agents will never connect to other *Product* agents. In the social simulation, each agent interacts with another agent in a pair-wise fashion that is modeled as a Poisson process with rate 1, independent of all other agents. By assuming a Poisson process of interaction, we are claiming that there is at most one interaction at any given time. Here, the probability of interaction between agents a_i and a_j is shown by p_{ij} and is defined as a fraction of the connection weight between these agents over the total connections that agent i makes with the other agents. Therefore,

$$p_{ij} = \begin{cases} \frac{e_{ij}}{d^i_{out}} & i, j \in A_R \\ \frac{u_{ji}}{Threshold} & i \in A_R, j \in A_P \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where d^i_{out} is the out-node degree of a *Regular* agent i and the *Threshold* parameter is the total number of links that *Product* agent can make with *Regular* agents. The bounds on *Threshold* are a natural consequence of the limited budget of companies in advertising their products.

At each interaction there is a chance for agents to influence each other and change their desire vector for purchasing or consuming a product. In all these interactions *Product* agents, the immutable agents, are the only agents who do not change their attitude and have a fixed desire vector. The probability that agent j influences agent i is denoted as α_{ij} and is calculated based on the out-node degree of agent j as:

$$\alpha_{ij} = \begin{cases} \frac{e_{ji}}{d^j_{out}} & i, j \in A_R \\ cte & i \in A_R, j \in A_P \end{cases} \quad (3)$$

Figure 2 shows a simple example of how to calculate p_{ij} and α_{ij}.

The other important parameter in the agent influence process is ε_{ij}, which determines how much agent j will influence agent i. This parameter is derived from a Gaussian distribution assigned to the membership group of agent j based on the experience of agent i with this group. Therefore, this value can easily be extracted from the previously defined vector $\overrightarrow{S_i}$.

As a final note, in this model the agents can access the following information:

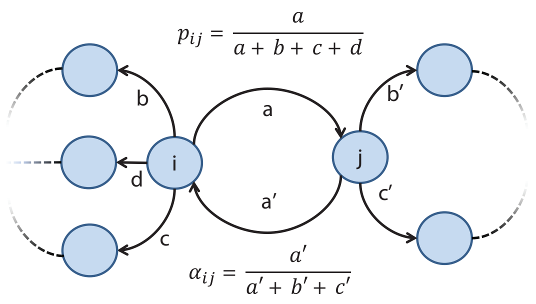

$p_{ij} = \frac{a}{a+b+c+d}$
b
a
b′
d
i
j
a′
c′
c
$\alpha_{ij} = \frac{a'}{a'+b'+c'}$

Figure 2: An illustration of how the probability of interaction (p) and the probability of influencing others (α) is calculated between the *Regular* agents.

1. the links connecting agents that possess a history of past interactions. Each agent is aware of its connections with neighbors and their weights;
2. the group membership of neighboring agents and other select members of the community.

The ultimate goal of our marketing problem is to recognize the influential agents in the graph and define $u_{ji}s$ in a way to get the maximum benefit of the product advertising.

3. NETWORK GENERATION

To evaluate the performance of our proposed optimization method on identifying influential agents in a variety of networks, we simulate the creation of agent networks formed by the combined forces of homophily and group membership. Since social communities often form a scale-free network, whose degree distribution follows a power law [5], we model our agent networks using the network generation method described in [25]. Note that this network only connects the regular agents $(a_i \in A_R)$. The connection between the *Product* and *Regular* agents is identified later in a way to optimize the efficiency of the product marketing.

Following the network data generation method in [23], we control the link density of the network using a parameter, *ld*, and value homophily between agents using a parameter, *dh*. The effects of value homophily are simulated as follows:

1. At each step, a link is either added between two existing nodes or a new node is created based on the link density parameter (*ld*). In general, linking existing nodes results in a higher average degree than adding a new node.
2. To add a new link, first, we randomly select a node as the source node, a_i, and a sink node, a_j $(a_i, a_j \in A_R)$, based on the homophily value (*dh*), which governs the propensity of nodes with similar group memberships to link. Node a_j is selected among all the candidate nodes in the correct group, based on the degree of the node. Nodes with higher degree have a higher chance to be selected.
3. If a prior link exists between agent a_i and a_j, selecting them for link formation will increase the weight of their link by one.

Group membership also governs the process of reciprocal link formation. Once the link generation process starts and the source and sink nodes have been selected, we add a directed link from node a_i to node a_j by default, under the assumption that the first selected agent initiated the interaction. The group value judgment of the second node governs whether a reciprocal link is formed or not. We use an evaluation function $F_a(S)$ to map an observed group value S to a binary evaluation of interaction (positive or negative). We assume that all agents use the same evaluation function, which is:

$$F_a(S) = \begin{cases} 1: & S \geq 0.5 \\ -1: & S < 0.5 \end{cases}$$

The result of this process is to create clusters of agents with the same group labels within the network, since group membership affects both the probability of the initial interaction (through the homophily parameter) and also the reciprocal link formation.

To generate a new node, we first select a group label based on a uniform group distribution and assign that group label to the node. Then we add links between the new node and one of the existing nodes as we described above. The algorithm for generating the static network is outlined in Table 1.

Table 1: Agent Network Generator

```
Agent Network Generator (numNodes, numLabels, ld, dh)
i = 0
E = NULL
while $i <$ numNodes do
    sample $r$ from uniform distribution $U(0,1)$
    if $r \leq ld$ then
        connectNode(E,numLabels,dh)
    else
        addNodes(E,numLabels,dh)
        $i = i + 1$
    end if
end while
return E
```

4. ANALYSIS OF SYSTEM DYNAMICS

As explained in Section 2, the agent i's desire toward product p, is modeled as a random variable that assumes a scalar value after each interaction $(x_{ip} \in [-1\ 1])$. Therefore, since there exist P different products, each agent has a vector of random variables, $\overrightarrow{X_i}$, which indicates the desire of the agent toward all the available products in market. Following Hung et al. [12, 13], we model the desire dynamic of all agents as a Markov chain where the state of the system is a matrix of all agents' desire vectors at a particular iteration k and the state transitions are calculated probabilistically from the pair-wise interaction between agents connected in a network. The state of the system at the k^{th} iteration is a vector of random variables, denoted as $\mathbf{X}(k) \in \mathbb{R}^{NP\times 1}$ (created through a concatenation of N vectors of size P) and expressed as:

$$\mathbf{X}(k) = \begin{pmatrix} [\overrightarrow{X_1}(k)] \\ \vdots \\ [\overrightarrow{X_N}(k)] \end{pmatrix}$$

4.1 Interaction and Influence

In this work, we define interactions as any kind of information or belief sharing between two agents about the available products in the market. During these interactions, there is a possibility for one agent to influence the desire of the other one. As explained in Section 2, this possibility is modeled by parameter α_{ij} when agent i initiates the interaction with agent j. Also, in this interaction, we assume that the influenced agent will retain some fraction of its existing desire. This fraction is different for any single agent i while interacting with agent j, but remains fixed, and is denoted as $\varepsilon_{ij} \in [0\ 1]$. The dynamics of the model at each iteration k proceed as described in [13]:

1. Agent i initiates the interaction on a uniform probability distribution over all agents. Then agent i selects another agent among its neighbors with probability p_{ij}. Note that the desire dynamic can occur with probability $\frac{1}{N}(p_{ij} + p_{ji})$ as agent i's attitude can change whether it initiates the interaction or is selected by agent j.

2. Conditioned on the interaction of i and j:

   - With propagability α_{ij}, agent i will change its desire:
   $$\begin{cases} \overrightarrow{X_i}(k+1) = \varepsilon_{ij}\, \mathbf{M}\overrightarrow{X_i}(k) + (1-\varepsilon_{ij})\, \mathbf{M}\overrightarrow{X_j}(k) \\ \overrightarrow{X_j}(k+1) = \overrightarrow{X_j}(k) \end{cases} \tag{4}$$
   Recall that $\mathbf{M}$ is the pre-defined matrix indicating the correlation between the demands of different products.
   - With probability of $(1 - \alpha_{ij})$, agent i is not influenced by the other agent:
   $$\begin{cases} \overrightarrow{X_i}(k+1) = \overrightarrow{X_i}(k) \\ \overrightarrow{X_j}(k+1) = \overrightarrow{X_j}(k) \end{cases} \tag{5}$$

To analyze Equation 4 in detail, we rewrite the matrix calculation for agent i as follows:

$$\overrightarrow{X_i}(k+1) = \begin{pmatrix} \sum_{f=1}^{P} m_{1f}\,(\varepsilon_{ij}x_{if} + (1-\varepsilon_{ij})\, x_{jf}) \\ \vdots \\ \sum_{f=1}^{P} m_{Pf}\,(\varepsilon_{ij}x_{if} + (1-\varepsilon_{ij})\, x_{jf}) \end{pmatrix} \tag{6}$$

A closer look at each row of $(\overrightarrow{X_i}(k+1))$ reveals that the desire of agent i toward a product depends on own previous desire, a fraction of the other agent's desire toward that product, and the desire of both agents toward other available products in the market. This is an interesting result showing how our proposed model can express the complexity of real-world markets and capture the dependency of demand for different products [20].

4.2 Expected Long-term Desire

In this work, we determine the long-term desire of the agents for products in the system to find the optimized connection between the *Product* agents and *Regular* agents. In other words, we hypothesize that by examining the expected value of the steady state system $(\mathbf{X}(k))$, we are able to optimize the marketing strategy and identify the most influential nodes in the network. Therefore our goal in this section is to calculate the expectation vector of the system state since it captures all the interactions and the dependencies between the demand of the products.

The conditional expected value of the desire vector of agent i in a single pair-wise interaction between agents i and j, when the current state of the system is observed:

$$\begin{aligned} E[\overrightarrow{X_i}(k+1)|\mathbf{X}(k), j] &= (1-\alpha_{ij})\overrightarrow{X_i}(k) \\ &\quad + \alpha_{ij}\left[\varepsilon_{ij}\mathbf{M}\overrightarrow{X_i}(k) + (1-\varepsilon_{ij})\mathbf{M}\overrightarrow{X_j}(k)\right] \\ &= [\alpha_{ij}\varepsilon_{ij}\mathbf{M} + (1-\alpha_{ij})\mathbf{I}]\ \overrightarrow{X_i}(k) \\ &\quad + \alpha_{ij}(1-\varepsilon_{ij})\mathbf{M}\ \overrightarrow{X_j}(k) \end{aligned} \tag{7}$$

By defining matrix $\mathbf{W}(i,j) = \alpha_{ij}(1-\varepsilon_{ij})\,\mathbf{M}$, we rewrite Equation 7 in the form of:

$$\begin{aligned} E[\overrightarrow{X_i}(k+1)|\mathbf{X}(k), j] &= \overrightarrow{X_i}(k) + \mathbf{W}(i,j)\overrightarrow{X_j}(k) \\ &\quad - [\mathbf{W}(i,j) + \alpha_{ij}(\mathbf{I}-\mathbf{M})]\ \overrightarrow{X_i}(k) \end{aligned} \tag{8}$$

Therefore, based on the probability of interaction between two agents ($\frac{1}{N}(p_{ij} + p_{ji})$), the desire of *Regular* agents dynamically changes as specified in Equation 7. It is worthwhile to mention that matrix $\mathbf{W}$ is a factor of matrix $\mathbf{M}$, and it has the same dimensions of $P \times P$. Rewriting the dynamics of $\overrightarrow{X_i}$ in this way indicates that the desire vector of agent i at iteration $(k+1)$ is equivalent to its own desire plus the weighted desire of agent j at iteration k, minus its own weighted desire at that iteration. This finding shows that, in spite of having the extra matrix $\mathbf{M}$, extracted from the marketing situation, and a complicated notion of the agents' desire vector, the computation model simply follows [12], although the optimization approach must account for multiple product interactions.

We substitute $\mathbf{W}(i,j) + \alpha_{ij}(\mathbf{I}-\mathbf{M}) = \mathbf{S}(i,j)$, where $\mathbf{S}(i,j)$ again is dimension $P \times P$. Then, Equation 8 simplified as follows:

$$E[\overrightarrow{X_i}(k+1)|\mathbf{X}(k), j] = \overrightarrow{X_i}(k) - \mathbf{S}(i,j)\overrightarrow{X_i}(k) + \mathbf{W}(i,j)\overrightarrow{X_j}(k) \tag{9}$$

Next, we write the expected value of agent i's desire vector at iteration $(k+1)$ over all the possible interactions it initiates or is subject to by other agents' actions, conditioned on the state of the system at k. Recall that the interaction between i and j occurs with probability $\frac{1}{N}(p_{ij} + p_{ji})$.

$$\begin{aligned} E[\overrightarrow{X_i}(k+1)|\mathbf{X}(k)] &= \overrightarrow{X_i}(k) \\ &\quad - \sum_j \frac{1}{N}(p_{ij} + p_{ji})\ \mathbf{S}(i,j)\overrightarrow{X_i}(k) \\ &\quad + \sum_j \frac{1}{N}(p_{ij} + p_{ji})\ \mathbf{W}(i,j)\overrightarrow{X_j}(k) \end{aligned} \tag{10}$$

Now, we want to express the expected desire of all agents at iteration $(k+1)$ conditioned on all agents' previous desire. This step relies on both the laws of interacting expectations and linearity of expectations. Assembling a vector of all entries for each i results in:

$$E[\mathbf{X}(k+1)|\mathbf{X}(k)] = \mathbf{X}(k) + \mathbf{Q}\mathbf{X}(k) \tag{11}$$

where $\mathbf{Q}$ is a block matrix and each component of $\mathbf{Q} \in \mathbb{R}^{N\times N}$, considering Equation 10, is:

$$Q_{ij} = \begin{cases} \frac{1}{N}(p_{ij} + p_{ji})\mathbf{W}(i,j) & i \in A_R, j \in A \text{ and } i \neq j \\ -\frac{1}{N}\sum_j (p_{ij} + p_{ji})\mathbf{S}(i,j) & i \in A_R, j \in A \text{ and } i = j \\ +\frac{1}{N}(p_{ij} + p_{ji})\mathbf{W}(i,j) & \\ 0 & i \in A_P, j \in A \end{cases} \tag{12}$$

Finally, by calculating the expected value of Equation 11 and using the linearity of expectations, we have:

$$E[E[\mathbf{X}(k+1)|\mathbf{X}(k)]] = E[\mathbf{X}(k+1)] = E[\mathbf{X}(k)] + \mathbf{Q}\ E[\mathbf{X}(k)] \tag{13}$$

We define $\overrightarrow{\mu}_{\mathbf{X}}(k) \in \mathbb{R}^{NP\times 1}$ as the expected value vector of $\mathbf{X}(k)$. Therefore, the above equation is simplified as:

$$\overrightarrow{\mu}_{\mathbf{X}}(k+1) = \overrightarrow{\mu}_{\mathbf{X}}(k) + \mathbf{Q}\ \overrightarrow{\mu}_{\mathbf{X}}(k) \tag{14}$$

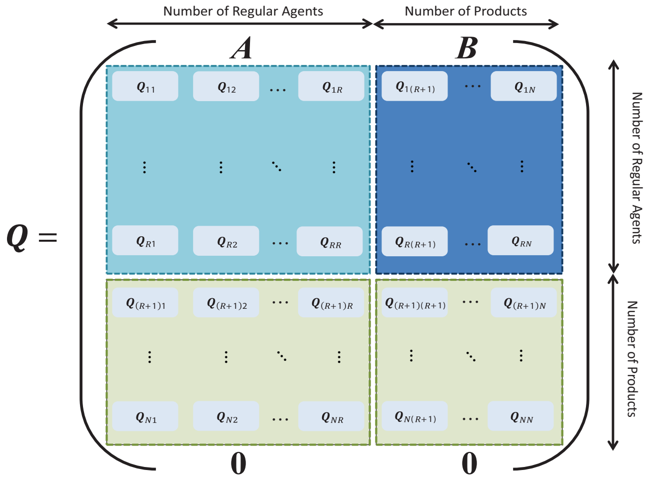


Figure 3: Q matrix is a block matrix with size $N \times N$ where N is the total number of agents $(R + P)$ and each block has the size of $P \times P$. Matrices A and B are the non-zero part of this matrix which represent the interactions among *Regular* agents and interactions between *Regular* agents and *Products*, respectively.

Since we are seeking the expected value of $\mathbf{X}(k)$ at steady state, the above equation when $k \to \infty$ reduces to:

$$\vec{\mu}_{\mathbf{X}}(\infty) = \vec{\mu}_{\mathbf{X}}(\infty) + \mathbf{Q}\,\vec{\mu}_{\mathbf{X}}(\infty) \Rightarrow \mathbf{Q}\,\vec{\mu}_{\mathbf{X}}(\infty) = 0 \quad (15)$$

In order to solve this system of equations efficiently, we decompose the matrices:

$$\mathbf{Q} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ 0 & 0 \end{pmatrix} \text{ and } \vec{\mu}_{\mathbf{X}}(\infty) = \begin{pmatrix} \vec{\mu}_{\mathbf{R}} \\ \vec{\mu}_{\mathbf{P}} \end{pmatrix} \quad (16)$$

Here $\mathbf{A} \in \mathbb{R}^{RP \times RP}$ is the sub-matrix representing the expected interactions among *Regular* agents while $\mathbf{B} \in \mathbb{R}^{RP \times P^2}$ represents the the expected interactions between *Regular* agents and *Product* agents. Figure 3 shows the breakdown of matrix $\mathbf{Q}$.

Moreover, $\vec{\mu}_{\mathbf{R}}$ and $\vec{\mu}_{\mathbf{P}}$ are vectors representing the expected long-term desire of *Regular* agents and *Product* agents, respectively, at iteration $k \to \infty$. Note that vector $\vec{\mu}_{\mathbf{P}}$ is known since the *Product* agents, the advertisers, are the immutable agents, who never change their desire. Solving for $\vec{\mu}_{\mathbf{R}}$ yields the vector of expected long-term desire for all regular agents, for a given set of influence-probabilities on a deterministic social network.

$$\mathbf{A}\,\vec{\mu}_{\mathbf{R}} + \mathbf{B}\,\vec{\mu}_{\mathbf{P}} = 0 \Rightarrow \vec{\mu}_{\mathbf{R}} = \mathbf{A}^{-1}(-\mathbf{B}\,\vec{\mu}_{\mathbf{P}}) \quad (17)$$

Now based on this analytical view of the system, we define an optimization method in following section to maximize the product sales through intelligent selection of the *Product* agent linkages.

5. NODE SELECTION METHOD

Using the analysis from the previous section, we can identify the influential nodes in the network and connect the products to those agents in a way that maximizes the long-term desire of the agents in the social system. Here, we define the objective function as the maximization of the weighted average of the expected long-term desire of all the *Regular* agents in the network toward all the products as:

$$max_u \sum_{1 \le k \le P} \sum_{i \in A_R} (\rho_i . \vec{\mu}_{\mathbf{R},i}) \quad (18)$$

$\vec{\mu}_{\mathbf{R},i}$ is the part of $\vec{\mu}_{\mathbf{R}}$ that belongs to agent i, and ρ_i parameter is simply a weight we can assign to agents based on their importance in the network. In the case of equivalent $\rho_i = 1$ for all the agents, the above function reduces to the arithmetic mean of the expected long-term desire vectors for all agents.

The goal of our proposed method is to assign a fixed number of *Product* agents with limited number of connections to a network of *Regular* agents in a way to optimize the objective function presented above. In Equation 17, matrix $\mathbf{A}$ and vector $\vec{\mu}_{\mathbf{P}}$ are known since the static network among the *Regular* agents and the fixed desire vector of the products are both known. We define the matrix $\mathbf{B}$ based on parameters of u_{ij}s. We substitute the probability of interaction, p_{ij}, occurring between agents i and j in matrix $\mathbf{Q}$, by Equation 2 of the model.

The partitioning of matrix $\mathbf{Q}$ in Equation 16 and the size of matrices $\mathbf{A}$ and $\mathbf{B}$ (Figure 3), indicates that the elements of matrix $\mathbf{B}$ are all off the diagonal. Therefore substituting the values of p_{ij} and p_{ji} of Equation 2 into Equation 12, $\mathbf{B}_{ij} = \frac{1}{N}\, u_{ji}\mathbf{W}(i,j) = \widehat{u} \otimes \mathbf{M}$. Here, $\widehat{u}$ contains all the variables and influence parameters and $\otimes$ indicates the Kronecker product [21].

Therefore, by rewriting Equation 17 as:

$$\vec{\mu}_{\mathbf{R}} = \mathbf{A}^{-1}[\widehat{u} \otimes \mathbf{M}]Vec(\widehat{\mu}_{\mathbf{P}}) \quad (19)$$

and using the following identity

$$[\widehat{u} \otimes \mathbf{M}]\; Vec(\widehat{\mu}_{\mathbf{P}}) = Vec(\mathbf{M}\,\widehat{\mu}_{\mathbf{P}}\,\widehat{u}),$$

Equation 17 becomes $\vec{\mu}_{\mathbf{R}} = \mathbf{A}^{-1}Vec(\mathbf{M}\,\widehat{\mu}_{\mathbf{P}}\,\widehat{u})$, which is solved using convex optimization methods. Therefore the optimal assignment of *Product* agents to *Regular* agents is obtained through the following optimization problem:

$$\begin{aligned} \underset{\widehat{u}}{\text{maximize}} \quad & \|\mathbf{A}^{-1}Vec(\mathbf{M}\,\widehat{\mu}_{\mathbf{P}}\,\widehat{u})\|_1 \\ \text{subject to} \quad & x_{ip} \in [-1\;1],\; \forall i \in A_R, \\ & \sum_{j \in A_R} u_{ij} = cte. \end{aligned} \quad (20)$$

To solve this optimization problem we used the CVX toolbox of Matlab which is useful for convex programming and minimized the dual of our objective function.

6. EVALUATION

6.1 Experimental Setup

We conducted a set of simulation experiments to evaluate the effectiveness of our proposed node selection method on marketing the items in a simulated social system with a static network. The parameters of the model for all the runs are summarized in Table 2(a). All the results are computed over an average of 30 runs with 100 *Regular* agents and 10 *Product* agents.

In this work, we model four long-lasting groups, $(G_1, \ldots, G_4)$, with different feature vector distributions in our social simulation. Moreover, a group value judgment, (S_i), assigned to each group, is drawn from Gaussian distribution. We assumed that the group model has been learned by agents based on their previous experiences, each agent has its own fixed value judgment toward each group of agents and that value has been selected based on the assigned Gaussian distribution of the model. Consequently, this group value judgment affects the connection of agents during the network generation phase, as we described before. Table 2(b) shows the mean and standard deviations of the Gaussian distributions assigned to each group. Note that the membership in each group is

Table 2: Parameter settings
(a) Experimental parameters

Parameter	Value	Descriptions
R	100	Number of *Regular* agents
P	10	Number of *Product* agents
$Threshold$	2	Number of links between P and R agents
ε	0.4	Influence factor between P and R agents
α	0.6	Probability of influence between P and R agents
$N_{Iterations}$	10000	Number of iterations
N_{Run}	30	Number of runs

(b) Group model

Group	Mean Value	StDev
G1	0.9	0.05
G2	0.6	0.15
G3	0.4	0.15
G4	0.3	0.1

permanent for all agents and cannot be changed during the course of one simulation.

In the *Regular* and *Product* agent interaction, parameters α and ε are fixed for any interaction and are presented in Table 2(a). We assume that these parameters can be calculated by advertising companies based on user modeling. The p_{ij} values for this type of interaction are calculated using Equation 2 and are parametric.

Finally, the remaining part of the social system setup is matrix $\mathbf{M}$, which models the correlation between the demand for different products. This matrix is generated uniformly with random numbers between $[0\ 1]$ and, as it has a probabilistic interpretation, the sum of the values in each row, showing the total demand for one item, is equal to one.

6.2 Results

We compare our optimization-based algorithm with a set of centrality-based measures commonly used in social network analysis for identifying influential nodes based on network structure [15]. The comparison methods are:

Degree Assuming that high-degree nodes are influential nodes in the network is a standard approach for social network analysis. Here, we calculated the probability of joining a *Regular* agent based on the out-degree of the agents and attached the *Product* agents according to preferential attachment. Therefore, nodes with higher degree had an increased chance of being selected as an advertising target.

Closeness This is another commonly used influence measure in sociology, based on the assumption that a node with short paths to other nodes has a higher chance to influence them. Here, we averaged the shortest paths of a node to all the other nodes in the network and sorted the nodes according to this measure. Nodes with shorter average path had a higher chance of being selected as a target.

Betweenness This centrality metric measures the number of times a node appears on the geodesics connecting all the other nodes in the network. Nodes with the highest value of betweenness had the greatest probability of being selected.

Random Finally, we consider selecting the nodes uniformly at random as a baseline.

To evaluate these methods, we started the simulation with an initial desire vector set to 0.02 for all agents, and simulated 10000

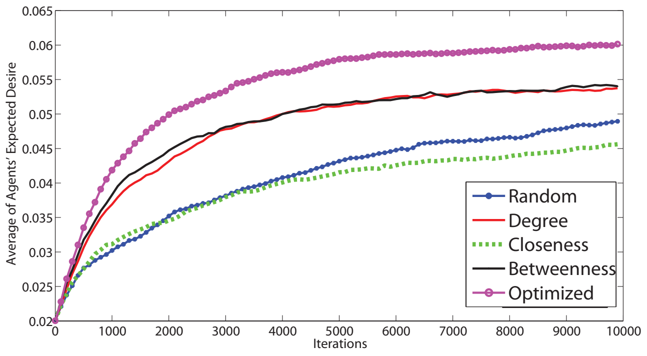


Figure 4: The average of agents' expected desire vs. the iterations. The average is across all the products and over 30 different runs. Our proposed method has the highest average in comparison to other methods which shows its capability as a method for targeted advertisement in a social system.

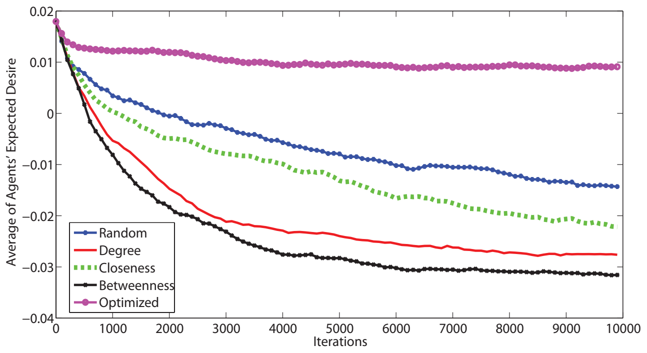


Figure 5: The average of agents' expected desire vs. iterations. In this simulation, the negative effect of advertising products against other products has been increased. This result demonstrates that our proposed method is more robust to the commonly occurring condition where increasing the desire toward one item has a higher negative effect on the desire of agent toward other products.

iterations of agent interactions. The entire process of interaction and influence is governed based on the previous formulas given in Section 4 and extracted parameters from the network. At each iteration, we calculated the average of the expected desire value of agents toward all products. Figure 4 shows this result for 100 agents and 10 advertisements. As explained before, the desire vector of *Product* agents are fixed for all products; in our simulation is was set to 1 for the product itself and -0.05 for all other products (e.g., $\mu_2 = [-0.05\ 1\ \ -0.05 \ldots\ \ -0.05])$. The results for this condition show that the proposed method creates a higher total product desire in the social system and is more successful than other methods at selecting influential nodes. To test the robustness of our algorithm we modified the desire vector of *Product* agents and increased the negative effect of advertisements over other products by factor of three (e.g., $\mu_2 = [-0.15\ 1\ \ -0.15 \ldots\ \ -0.15]$). The result of this simulation is shown in Figure 5. We can see that in this case the average desire of agents has dropped dramatically for all methods except the proposed algorithm. Even in the cases of having high negative effect toward other products, this algorithm can adapt the node selection in a way to keep the desire of agents high and sell more products.

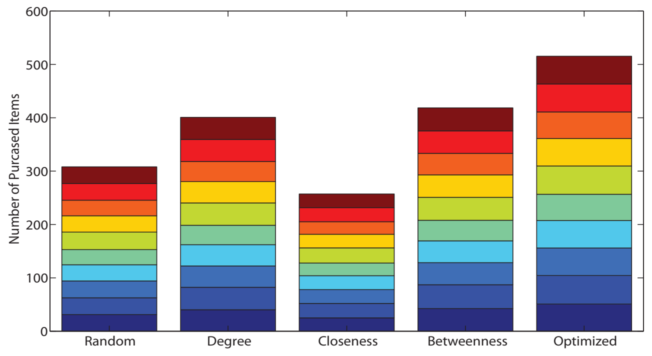


Figure 6: The number of sold items vs. different advertising methods. The assumption is that an agent with expected desire greater than 0.01 will purchase the product. Different colors in each bar indicates the number of sold items of each advertised products. As there exist ten different products, the bar is divided into ten parts.

To estimate the performance of algorithms in selling the products to *Regular* agents, we assumed that agents with expected desire higher than a threshold will purchase the product. Figure 6 shows the average of total purchased items by agents with the purchasing threshold as 0.01. Again, we see that our proposed algorithm is the most successful method in advertising and selling products.

7. RELATED WORK

Social ties between users play an important role in dictating their behavior. One of the ways this can occur is through social influence where a behavior or idea can propagate between friends. In [1], the authors examine the statistical correlation between the actions of friends in a social network by considering factors such as homophily and possible unobserved confounding variables. Hence it follows that it is not only important to advertise to your customer but also to your potential customer's friends.

One theory about influential nodes is that they can be characterized as a set of initial nodes that trigger behavior cascades. This set of nodes can then be identified either using probabilistic approaches [2, 18] or optimization-based techniques. For example, in [16] the behavior spreads in a cascading fashion according to a probabilistic rule, beginning with a set of initially active nodes. To identify influential agents, they select a set of individuals to target for initial activation, such that the cascade beginning with this active set is as large as possible in expectation. [18] find influential nodes in a complex social network by formulating the likelihood for information diffusion data, the activation time sequence data over all nodes; they propose an iterative method to search for the probabilities that maximize this likelihood.

Apolloni et al. [2] examine the spread of information through personal conversations by proposing a probabilistic model to determine whether two people will converse about a particular topic based on their similarity and familiarity. Similarity is modeled by matching selected demographic characteristics, while familiarity is modeled by the amount of contact required to convey information. On the other hand, [22] propose a learning method for ranking influential nodes and perform behavioral analysis of topic propagation; they compare the results with conventional heuristics that do not consider diffusion phenomena.

In this paper, we present one approach for framing and solving the optimization problem using convex programming. The optimization problem can also be solved using greedy algorithms (e.g., [19, 17]) that find approximate solutions using graph theory. [15] also utilized greedy algorithms to identify the influential nodes. Intelligent heuristics can be used to improve the scalability of influence maximization [8]. [9] made improvements upon existing greedy algorithms to further reduce run-time and proposed new degree discount heuristics that improve influence spread.

The effects of network topology on influence propagation have been studied in several domains, including technology diffusion, strategy adaption in game-theoretic settings, and the admission of new products in the market [15]. It has been demonstrated that the way information spreads is affected by the topology of the interaction network [26] and also that there exists a relationship between a person's social group and his/her personal behavior [24].

Social network analysis has been used as a tool for implementing effective viral marketing; influential nodes are identified either by following interaction data or probabilistic strategies. For example, Hartline et al. [11] solve a revenue maximization problem to investigate effective marketing strategies. The assumption is that a set of influential nodes can propagate the information about the items to other nodes, and therefore the objective is to find influential buyers in a social network. [27] presented a targeted marketing method based on the interaction of subgroups in social network. Moreover, [4], similar to this work, considered the existing homophily in social networks. But instead of finding influential nodes, they base their advertising strategy on the profile information of users and user-only models.

Our work differs from related work in that, our model not only considers homophily and group membership but also incorporates other important factors such as the positive and negative effect of competing product advertisements and the correlation among demand for different products. Our optimization approach is largely unaffected by the additional complexity since these factors only affect the long-term expected value and not the actual solution method. Outside of social network marketing approaches, there exist many marketing methods based on personalization techniques for delivering advertisements [14] or news [3].

8. CONCLUSION AND FUTURE WORK

Agent-based simulation is an important tool for understanding the behavior of social systems, enabling the analysis of population-level effects over long time horizons that are not easily studied within the confines of the lab. In this paper, we present a model for researching techniques of large-scale persuasion; rather than focusing on the content of the message, we address the problem of identifying agents that have a high probability of indirectly influencing a large number of additional agents. In popular parlance, these influential agents can be considered to be some mixture of Gladwell's connectors, mavens, and salesman who affect the long-term beliefs of other agents through their actions or communications [10]. Although this problem generalizes to domains such as politics and social media distribution, we demonstrate our method in a market scenario in which product representatives are attempting to maximize their sales through effective advertisement placement. To solve this influence maximization problem, we compute the steady-state expectation of the system and introduce a new optimization approach for selecting influential nodes. Contrary to previous work, our approach is suitable for computing an optimal solution across multiple products and handling the interactions of multiple influencing agents. Our results show that our technique conclusively outperforms a set of centrality-based techniques at selecting influential agents and maximizing total product sales. Our social model accurately reflects many of the complexities of real-world human systems, such as group membership effects, product preference dependencies, and a network structure driven by multiple conflicting forces. Moreover, in our model, an regular agent need not represent an individual person but can be thought of as an abstraction over communities of people. Then larger systems can simply be solved in a hierarchical, but non-exact, fashion. In our future work, we plan to mathematically analyze the short time behavior of the system. This analysis will allow us to generalized our solution technique to a dynamic network whose structure varies over time [6] which is a characteristic possessed by certain real-world social systems.

9. ACKNOWLEDGMENTS

This research was funded by AFOSR YIP FA9550-09-1-0525.

10. REFERENCES

[1] A. Anagnostopoulos, R. Kumar, and M. Mahdian. Influence and correlation in social networks. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 7–15. ACM, 2008.

[2] A. Apolloni, K. Channakeshava, L. Durbeck, M. Khan, C. Kuhlman, B. Lewis, and S. Swarup. A study of information diffusion over a realistic social network model. In *2009 International Conference on Computational Science and Engineering*, pages 675–682. IEEE, 2009.

[3] L. Ardissono, L. Console, and I. Torre. On the application of personalization techniques to news servers on the www. *AI* IA 99: Advances in Artificial Intelligence*, pages 261–272, 2000.

[4] A. Bagherjeiran and R. Parekh. Combining behavioral and social network data for online advertising. In *IEEE International Conference on Data Mining Workshops (ICDMW'08)*, pages 837–846. IEEE, 2008.

[5] A. Barabasi and E. Bonabeau. Scale-free networks. *Scientific American*, pages 60–69, May 2003.

[6] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *Arxiv preprint arXiv:1012.0009*, 2010.

[7] M. Cha, A. Mislove, and K. Gummadi. A measurement-driven analysis of information propagation in the flickr social network. In *Proceedings of the 18th International Conference on World Wide Web*, pages 721–730. ACM, 2009.

[8] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1029–1038. ACM, 2010.

[9] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 199–208. ACM, 2009.

[10] M. Gladwell. *The tipping point: How little things can make a big difference*. Little, Brown and Company, 2000.

[11] J. Hartline, V. Mirrokni, and M. Sundararajan. Optimal marketing strategies over social networks. In *Proceeding of the 17th International Conference on World Wide Web*, pages 189–198. ACM, 2008.

[12] B. Hung. Optimization-based selection of influential agents in a rural afghan social network. *Master's Thesis, Massachusetts Institute of Technology*, 2010.

[13] B. Hung, S. Kolitz, and A. Ozdaglar. Optimization-based influencing of village social networks in a counterinsurgency. In *Proceedings of the 4th International Conference on Social Computing, Behavioral-cultural Modeling and Prediction*, pages 10–17. Springer-Verlag, 2011.

[14] P. Kazienko and M. Adamski. Adrosa–adaptive personalization of web advertising. *Information Sciences*, 177(11):2269–2295, 2007.

[15] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146. ACM, 2003.

[16] D. Kempe, J. Kleinberg, and É. Tardos. Influential nodes in a diffusion model for social networks. *Automata, Languages and Programming*, pages 1127–1138, 2005.

[17] M. Kimura and K. S. nd R. Nakano. Extracting influential nodes for information diffusion on a social network. In *Proceedings of the National Conference on Artificial Intelligence*, 2007.

[18] M. Kimura, K. Saito, R. Nakano, and H. Motoda. Finding influential nodes in a social network from information diffusion data. *Social Computing and Behavioral Modeling*, pages 1–8, 2009.

[19] M. Kimura, K. Saito, R. Nakano, and H. Motoda. Extracting influential nodes on a social network for information diffusion. *Data Mining and Knowledge Discovery*, 20(1):70–97, 2010.

[20] J. Leskovec. Amazon product co-purchasing network. In *http://snap.stanford.edu/data/amazon0302.html.*, 2003.

[21] J. R. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics, 2nd Ed.* Wiley, 1999.

[22] K. Saito, M. Kimura, K. Ohara, and H. Motoda. Behavioral analyses of information diffusion models by observed data of social network. *Advances in Social Computing*, pages 149–158, 2010.

[23] P. Sen and L. Getoor. *Link-based classification*. Technical Report, CS-TR-4858, University of Maryland, Reading, Massachusetts, 2007.

[24] P. Singla and M. Richardson. Yes, there is a correlation:-from social networks to personal behavior on the web. In *Proceeding of the 17th International Conference on World Wide Web*, pages 655–664. ACM, 2008.

[25] X. Wang, M. Maghami, and G. Sukthankar. Leveraging network properties for trust evaluation in multi-agent systems. In *Proceeding of the 2011 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Lyon, France*, 2011.

[26] F. Wu, B. Huberman, L. Adamic, and J. Tyler. Information flow in social groups. *Physica A: Statistical and Theoretical Physics*, 337(1-2):327–335, 2004.

[27] W. Yang, J. Dia, H. Cheng, and H. Lin. Mining social networks for targeted advertising. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences-Volume 06*, pages 137–1. IEEE Computer Society, 2006.

Predicting Your Own Effort

David F. Bacon
IBM Research
dfb@watson.ibm.com

Yiling Chen
SEAS, Harvard University
yiling@eecs.harvard.edu

Ian Kash
MSR Cambridge
iankash@microsoft.com

David C. Parkes
SEAS, Harvard University
parkes@eecs.harvard.edu

Malvika Rao
SEAS, Harvard University
malvika@eecs.harvard.edu

Manu Sridharan
IBM Research
msridhar@us.ibm.com

ABSTRACT

We consider a setting in which a worker and a manager may each have information about the likely completion time of a task, and the worker also affects the completion time by choosing a level of effort. The task itself may further be composed of a set of subtasks, and the worker can also decide how many of these subtasks to split out into an explicit prediction task. In addition, the worker can learn about the likely completion time of a task as work on subtasks completes. We characterize a family of scoring rules for the worker and manager that provide three properties: information is truthfully reported; best effort is exerted by the worker in completing tasks as quickly as possible; and collusion is not possible. We also study the factors influencing when a worker will split a task into subtasks, each forming a separate prediction target.

Categories and Subject Descriptors

J.4 [**Social and Behavioral Sciences**]: Economics

General Terms

Economics, Theory

Keywords

Information Elicitation, Proper Scoring Rule, Principal-Agent

1. INTRODUCTION

Software engineering is one of many domains with complex and modular tasks. There are often information asymmetries, both between the worker performing a task and the manager supervising and between the two of them and the rest of the organization or company. In such environments, it is important for the organization to be able to elicit accurate predictions from worker-manager teams in regard to when individual tasks are expected to complete. By eliciting accurate predictions, this enables good decision-making in regard to scheduling resources to projects (such as bug fixes or new features), and in regard to coordination of projects.

A particular challenge is that a worker with information relevant to the prediction task also controls the completion time through the amount of effort exerted on the task. In modeling this, we consider a single worker and a single manager. The worker works on a sequence of tasks and both the worker and the manager receive a score based on predictions and completion times for each task completed. We assume that the organization (or company) couples the score received by the worker or manager with incentives, be they social-psychological such as praise or visibility, or material rewards through prizes or the payment of bonuses. Based on this, we assume that the worker and the manager each seek to maximize total expected score.

The role of the worker is to share information relevant to the expected completion time of the task with the manager, in order to enable accurate predictions, and also to decide on whether to work at "best effort" or less than best effort. The role of the manager is to combine information received from the worker with her own information (if any), and make accurate predictions to the organization regarding the completion time of tasks. We tackle the issue of how to elicit truthful information and thus accurate predictions from the worker and manager, as well as how to elicit best effort from the worker.[1]

In essence, our problem is a combination of a repeated principal-agent problem and a prediction problem. In a principal-agent setting, a principal wishes to elicit a desired effort level from an agent but does not require the agent to make any predictions. On the other hand in a prediction problem, accurate predictions of the outcome of an event are sought but without considering that the distribution on outcomes might be something that can be controlled by the agent doing the prediction. In contrast we seek to establish both accuracy and the investment of best effort.

Our main technical result is a characterization of a class of scoring rules that are able to align incentives with both accurate prediction and the investment of best effort. In addition, the scoring rules inherently preclude the possibility of collusion between the worker and manager in their participation in the scoring system. For example, it is not useful for a manager and worker to agree that the worker will deliberately slow down in return for a prediction task with lower variance and thus the potential for higher total score to the worker-manager pair.

[1]We assume that existing incentive schemes within the organization (e.g., pay, promotion, etc.) encourage best effort work, all things being equal. For this reason, it is sufficient for our purposes that the incremental incentives provided by the scoring scheme, work with (not against) best effort. In particular, we want to preclude working at less than best effort leading to a higher expected score.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

In addition, we consider the effect of a scoring system on whether or not a worker will choose to split a task into multiple prediction targets. For this purpose, we model a task as a sequence of subtasks, where a subtask is conceptualized as a unit of work with a well-defined end point, and for which the time to complete the unit of work may be informative as to the time to complete other subtasks that comprise a task. With this in mind, we study the incentives for a worker to "split-out" a subtask for the purpose of a separate prediction target.[2] The qualitative result we obtain is that there is a greater propensity to split subtasks for which the completion times are positively correlated than those for which the completion times are independent. A simulation study completes the paper, providing a quantitative analysis of the trade-off between the frequency of "splitting" prediction into subtasks, the degree to which the distribution on subtask completion time is correlated, and a parameterization of the scoring rule that affects how much payment is made per subtask target vs how much payment must be made in catch-up upon the completion of a task.

1.1 Related Work

Scoring rules have been developed to measure the performance of experts who are solicited to reveal their probability assessments regarding uncertain events. They have been used in a variety of scenarios, from weather forecasting to prediction markets [3, 5, 4, 7]. *Proper* scoring rules incentivize truthful reporting of likelihood estimates. An overview of the theory behind proper scoring rules can be found in Gneiting and Raftery [3].

Proper scoring rules typically require that the outcome of the uncertain event will be revealed and the agent whose assessment is elicited can not influence the outcome. In our setting, the prediction of effort required to complete a task and the outcome or realized effort are not independent; both are influenced by the worker. Shi *et al.* [11] consider situations where agents may be able to take actions that influence the outcome. They propose *principal-aligned* mechanisms that do not incentivize agents to take actions that reduce the utility of the principal. Their setting considers eliciting a probability distribution and the outcome space is discrete. Our setting allows for continuous effort level and we seek to elicit the expectation as well as incentivize best effort. The result of Shi *et al.* [11] can be generalized to the setting of eliciting the expectation for a random variable over a continuous outcome space using the characterization of Savage [10], which is also used to derive our characterization in Section 3. With this generalization, it is possible to derive our Theorem 7 by assigning a particular utility function to the principal and applying the result of Shi *et al.* [11]. However, this approach seems unnecessarily complicated in our setting, and we derive our results by directly considering desirable properties of the incentive mechanism.

There is a vast literature on *principal-agent* models [2, 6]. In a classical principal-agent model with hidden action, an agent chooses an action to take that is costly for him

[2]Our viewpoint is that it is the worker, not the manager, who is privy to information in regard to subtasks. Moreover, we can imagine situations in which predictions in regard to subtasks rather than in regard to the aggregate time for a task is useful; e.g., for sharing information with other workers, for re-planning, and in order to collect data to enable the training of predictive models in order to enable better organizational efficiency going forward.

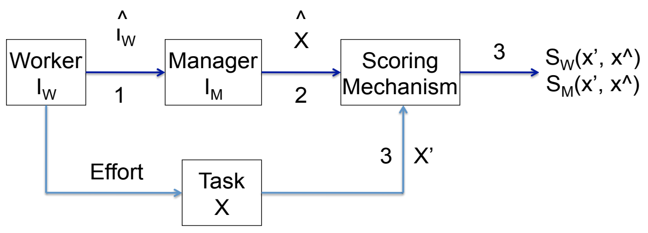


Figure 1: Timeline of the worker-manager game.

but beneficial for the principal in exchange for a promise of payment. The principal cannot directly observe the agent's action, but the stochastic correlation between actions and outcomes (that is, the probability of observing an outcome given that the agent takes an action), is common knowledge. For example, the agent's action can be a level of effort exerted with the probability of success for a project an increasing function of the level of effort. Knowing the stochastic correlation, the principal seeks to incentivize the agent to take a desirable action using contracts with payments based on the outcome.

Radner [9] considers an infinitely repeated setting for the principal-agent problem. In Radner's setting, the game is composed of sequences of review and penalty periods. By allowing the players' actions in one period to depend on the history of previous periods, the principal can observe the results of the agent's actions and *punish* the agent if the agent's performance fails some statistical test of efficiency. Radner shows that for high enough discount factors, there exist equilibria, consisting of reward-decision pairs, of the infinitely repeated game that are strictly more efficient than the short-term equilibrium. Our setting is different in that it combines the challenge of eliciting desirable actions with that of eliciting information from an agent. Our setting introduces information asymmetry about the stochastic correlation between the action and the outcome, allowing the agent to have private information about this stochastic correlation. The principal would like to elicit the information from the agent so as to obtain a better prediction, which is then used by the principal to set the reward for the agent. Because the reward of the agent now depends on the reported information, this introduces incentives to lie about the information or act in a suboptimal way. Given this tension, we aim to achieve truthful elicitation of private information as well as elicitation of the desirable action.

2. THE BASIC MODEL

We consider the incentive design problem for a company (the principal), whose goal is to truthfully elicit information from its employees as well as incentivize them to exert optimal level of effort. The basic model considers a single task with two agents, a worker and a manager, each with private information in regard to likely completion time of the task. The worker shares information with the manager, who then combines this information with his own private information and makes a prediction. The worker then exerts effort, and at some subsequent time the task completes and the worker informs the system (and the manager) of this event. We assume that only a truly completed task can be claimed as complete, but allow a worker to reduce effort below best effort, including to pretend a task is not complete when it has been completed. Eventually, a score is assigned to both the worker and the manager. Later, we extend the basic model to include structure in regard to subtasks and also to consider a sequence of tasks.

Let X denote the random variable for the time the task takes to complete under the best effort by the worker. Assume that the realized value of X is nonnegative and upper bounded, i.e. $x \in (0, x_{\max})$. Neither the manager nor the worker knows the realization of X. But they each have some private information, denoted as random variables I_m and I_w respectively, on the completion time under best effort. The joint distribution $\Pi(X, I_m, I_w)$ is common knowledge to them, but not known to the company. Assuming Π is not known to the company ensures a broad range of priors are considered possible. In particular, this allows $E[X|I_m]$, $E[X|I_w]$, and $E[X|I_m, I_w]$ all take on all values in $(0, x_{\max})$. This ensures that rules we derive work for a broad range of beliefs, similar to proper scoring rules requiring truthful reporting be optimal for all probability distributions. If the company believes that only a significantly restricted set of priors is possible, there may be additional rules that our results do not characterize. Note also that these expectations are well defined because X is bounded.

The manager and the worker play a three-stage game as shown in Figure 1. In stage 1, the worker can communicate with the manager and share information. In stage 2, the manager makes a prediction $\hat{x}$ about the completion time of the task under the worker's best effort. In stage 3, the worker exerts some effort and completes the task in time x'. While the worker cannot exert more than his best effort and complete the task in time less than x, he can work at a more slack pace and take time $x' > x$ to complete the task. However, we require that $x' \leq x_{\max}$ because otherwise it will be clear to both the manager and company that he is not working efficiently.

We assume that both the manager and the worker are risk neutral. We further assume that the worker, all things being equal, is indifferent between working at best effort or "slacking." In other words, if the worker can get a higher expected score through a best-effort strategy rather than slowing down, then this is the approach the worker will take. Our results also hold when there is an existing, strict incentive for best effort over slacking, for example because of existing incentives in the company.

We consider incentive mechanisms (we refer to them as *scoring systems*) that reward the manager and the worker based on the manager's prediction of the completion time and the worker's actual completion time. At the end of stage 3, a manager is rewarded according to the score $S_m(x', \hat{x})$ and the worker according to the score $S_w(x', \hat{x})$. We require S_m and S_w to be differentiable with respect to x' and $\hat{x}$. The goal of a scoring system is to incentivize the report of an accurate prediction at best effort and the exertion of the best effort.

2.1 Desirable Properties of Scoring Systems

Our model is a simple two-player three-stage game. We hence consider the perfect Bayesian equilibrium of the game and desire good behavior of the manager and the worker at the equilibrium. The following are four properties we would like a scoring system to achieve at the equilibrium:

1. **Information sharing**. For all Π, the worker shares his private information I_w honestly with the manager in stage 1.
2. **Report the mean**. For all Π, when estimating the time required to complete a task under best effort of the worker, the manager's optimal report in stage 2 is $\hat{x} = E[X|I]$ where I is all information available to the manager at the time, given equilibrium beliefs.
3. **Best effort**. For all $\hat{x}$, it is optimal for the worker to exert his best effort and choose $x' = x$ for all realizations x in stage 3.
4. **Collusion-proofness**. For all Π, the total expected score of the manager and the worker is maximized by reporting $\hat{x} = E[X|I_w, I_m]$ and exerting best effort such that $x' = x$ for all realizations x.

If the above four properties are satisfied, we will have a perfect Bayesian equilibrium where the worker shares all his information with the manager, the manager truthfully reports his expectation of the completion time under best effort given both pieces of information, and the worker completes the task as quickly as possible. Moreover, this equilibrium is collusion-free, such that no joint deviation can lead to an increase in the total expected score.

3. CHARACTERIZATION OF SCORING SYSTEMS

We proceed to characterize scoring systems that satisfy our desirable properties. The main technical challenge is to simultaneously address the need for accurate prediction and retain incentives for the worker to adopt best effort.

First, we consider the best effort property. It's easy to see that if choosing $x' = x$ is optimal for the worker given any x and prediction $\hat{x}$, the worker's score $S_w(x', \hat{x})$ must be a decreasing function of x'.

OBSERVATION 1. *A scoring system satisfies best effort if and only if* $\frac{\partial S_w(x', \hat{x})}{\partial x'} \leq 0$.

For example, a simple scoring rule $S_w(x', \hat{x}) = 2\hat{x} - x'$ can incentivize the worker to exert his best effort.

Given the best effort property, we know that x' is set to x at the equilibrium. The report the mean property requires a scoring system to incentivize the manager to honestly report her expected completion time given all available information. This is exactly the problem addressed by proper scoring rules for eliciting the mean of a random variable. Proper scoring rules for eliciting the mean of a random variable satisfy the property that reporting the mean maximizes expected score. Hence, we have an immediate solution based on the definition of proper scoring rules.

OBSERVATION 2. *If the best effort property is satisfied, the scoring system satisfies the report the mean property if and only if* $E(X|I) \in \operatorname{argmax}_{\hat{x}} E(S_m(X, \hat{x})|I)$.

We can use any proper scoring rule as the manager scoring rule, in conjunction with a worker scoring rule that incentivizes best effort, to achieve the report the mean property. For example, $S_m(x', \hat{x}) = b - (\hat{x} - x')^2$ for an arbitrary parameter b uses a quadratic scoring rule.

While it is easy to achieve both best effort and report the mean properties at an equilibrium, satisfying information sharing and collusion-proofness is less straightforward.

Consider the pair of the worker and manager scoring rules mentioned above, $S_w(x', \hat{x}) = 2\hat{x} - x'$ and $S_m(x', \hat{x}) = b - (\hat{x} - x')^2$. The worker may not want to share his information with the manager if his information will lead to a lower prediction $\hat{x}$ by the manager. In addition, the total score can be increased if the worker and the manager collude. To see this, note that the manager can report a larger prediction and the worker can work slowly to perfectly match the manager's prediction, which increases the worker's score while maximizing the manager's score. Below, we characterize the conditions for achieving all four desired properties simultaneously.

3.1 A Family of Scoring Rules

We first consider how to satisfy the information sharing property. This will require that the worker is also rewarded for a more accurate prediction.

LEMMA 3. *If the best effort and report the mean properties are satisfied, the information sharing property is satisfied if and only if $E(X|I) \in \operatorname{argmax}_{\hat{x}} E(S_w(X, \hat{x})|I)$.*

PROOF. The worker can influence the prediction $\hat{x}$. In an extreme case, when all relevant information is possessed by the worker, the prediction is effectively made by the worker. In order for the worker to predict the mean, the worker scoring rule needs to be a proper scoring rule for the random variable X. Because $E(X|I)$ maximizes a worker's score given any information set I, for any I_m and I_w, $E(X|I_w, I_m)$ maximizes the worker's expected score $E(S_w(X, \hat{x})|I_w, I_m)$. Hence, the worker is better off sharing the information with the manager to have the manager report $E(X|I_w, I_m)$. □

Next, we consider achieving collusion-proofness. Let $S_T(x', \hat{x})$ denote the sum of the worker and manager scores. If the manager and the worker collude to report a prediction $\hat{x}$ and complete the task in time x', collusion-proofness requires that the manager-worker pair is incentivized to report the mean and exert best effort. These are analogous to achieving information sharing and best effort when the worker has all information and the manager has no information. Let $S_T(x', \hat{x}) = S_w(x', \hat{x}) + S_m(x', \hat{x})$ be the total scoring rule. The following result follows immediately.

LEMMA 4. *Collusion-proofness is satisfied if and only if $\frac{\partial S_T(x', \hat{x})}{\partial x'} \leq 0$ and $E(X|I) \in \operatorname{argmax}_{\hat{x}} E(S_T(X, \hat{x})|I)$.*

This means that if a scoring system satisfies best effort, report the mean, and information sharing we essentially get collusion-proofness for free with the mild additional condition that the total scoring rule also satisfies best effort (a sufficient condition for which is that the manager's scoring rule satisfies best effort). Combining the results characterizes scoring systems that satisfy all four desirable properties.

LEMMA 5. *A manager-worker scoring system satisfies information sharing, report the mean, best effort, and collusion-proofness at a perfect Bayesian equilibrium if and only if the following conditions are satisfied:*

- $\frac{\partial S_w(x', \hat{x})}{\partial x'} \leq 0$.
- $\frac{\partial S_T(x', \hat{x})}{\partial x'} \leq 0$.
- $E(X|I) \in \operatorname{argmax}_{\hat{x}} E(S_m(X, \hat{x})|I)$.
- $E(X|I) \in \operatorname{argmax}_{\hat{x}} E(S_w(X, \hat{x})|I)$.

for all information sets I.

Intuitively, Lemma 5 requires that the worker score and the manager score are all given by a proper scoring rule for eliciting the mean (it is immediate that the total score must also be given by a proper scoring rule), in addition to the worker and total scores being a decreasing function of the actual completion time. For example, $S_w(x', \hat{x}) = S_m(x', \hat{x}) = f(x') + 2cx'\hat{x} - c\hat{x}^2$, where $f'(x') + 2c\hat{x} < 0$ and $c > 0$ is a family of scoring systems that satisfy all four desirable properties. A theorem due to Savage [10] characterizes all (differentiable) proper scoring rules for eliciting the mean.

THEOREM 6 (SAVAGE [10]). *For S differentiable in $\hat{x}$, $E(X|I) \in \operatorname{argmax}_{\hat{x}} E(S(X, \hat{x})|I)$ if and only if $S(x', \hat{x}) = f(x') + G(\hat{x}) + (x' - \hat{x})G'(\hat{x})$ where $E[f(X)|I]$ is finite for all Π and G is a differentiable convex function.*

Note that a sufficient condition for $E[f(X)|I]$ to be finite for all Π is that f is bounded on $(0, x_{\max})$. Combining Theorem 6 with Lemma 5 yields a more precise characterization.

THEOREM 7. *A manager-worker scoring system satisfies information sharing, report the mean, best effort, and collusion-proofness at a perfect Bayesian equilibrium if and only if the following conditions are satisfied:*

- *$S_w(x', \hat{x}) = f_w(x') + G_w(\hat{x}) + (x' - \hat{x})G'_w(\hat{x})$ where f_w is a differentiable function such that $E[f_w(X)|I_w]$ is finite for all Π and G_w is a differentiable convex function.*
- *$S_m(x', \hat{x}) = f_m(x') + G_m(\hat{x}) + (x' - \hat{x})G'_m(\hat{x})$ where f_m is a differentiable function such that $E[f_m(X)|I_m, I_w]$ is finite for all Π and G_m is a differentiable convex function.*
- *$f'_w(x') + G'_w(\hat{x}) \leq 0$ for all x',$\hat{x} \in (0, x_{\max})$.*
- *$f'_w(x') + f'_m(x') + G'_w(\hat{x}) + G'_m(\hat{x}) \leq 0$ for all x',$\hat{x} \in (0, x_{\max})$.*

Finally, note that this means we can derive a scoring system from a differentiable convex pair of Gs whose derivatives we can upper bound by taking $f'_w(x') = -|\sup_{\hat{x}} G'_w(\hat{x})|$ and similarly for f_m.

4. TASK DECOMPOSITION

Continuing, we now consider that a task has substructure, with a task represented as a series of subtasks. Based on this, we allow a worker-manager team to elect to split-off individual subtasks (or contiguous subtasks) to become identified prediction tasks in their own right; i.e., essentially partitioning the task into a distinct set of pieces, each of which has an associated prediction problem.

In increasing the realism of the model, we also situate the prediction task for a single task in the context of a repeated version of the problem, in which a worker has a sequence of tasks. In this context, the following property is useful:

5. **Always non-negative**. The score of the worker and the manager is always non-negative for all realizations of x and all reports $\hat{x}$.

If the score is always non-negative, then our best effort property immediately guarantees that best effort is also optimal for a worker facing a sequence of tasks, in that this will maximize both the total score for sequence of tasks and the score per unit time.[3]

This noted, we can focus back on a single task and introduce formalism to make precise what is intended by a subtask. Let $X = X_1 + \ldots + X_k$ denote a task X composed of k subtasks $X_1, \ldots, X_k$. The worker decides which sets of subtasks are to become targets of the scoring system. For example, the worker might prefer to make a single prediction, thereby being scored just once after completing the task in its entirety. Another option is that the worker may prefer to make k predictions (hence receiving k scores), one for each subtask. Alternately, the worker select subtask X_1 as a target, then subtask X_2, and then subtasks $X_3, \ldots, X_k$ aggregated into one chunk of work for the purpose of prediction. We assume that the degree to which the prediction problem associated with a task may be split-out into subtasks is knowledge that is private to the worker and *a priori* not known to the manager.

We allow the worker to make online decisions about which subtasks to split-out as separate prediction targets. That is, if the worker initially decides to get scored for X_1, after this is done he can then choose whether to next get scored for X_2 or instead to combine X_2 with some number of subsequent subtasks (we assume subtasks must be completed in order). As we are focusing on decisions made by the worker, we will only discuss S_w. The report the mean and collusion proofness properties can be retained through an appropriate choice of S_m. To be able to make concrete statements, we focus on the special case $S_w(x, \hat{x}) = f(x') + 2cx'\hat{x} - c\hat{x}^2$.

4.1 Independent Subtasks

For a simple model, consider a worker with two subtasks, denoted by random variables X_1 and X_2, and each with discrete support $\{a, b\}$, with $0 < a < b \leq 1$ and $x_{\max} = b$.

For this setting with two subtasks, the choice of the worker in regard to prediction targets is as follows:

- Adopt the complete task as a prediction target, share information in regard to $X = X_1 + X_2$ (with the manager making a prediction), work on them both, and then receive a score.
- Split-out X_1 as the first prediction target, share information with the manager (with the manager making a prediction), work on X_1 and receive a score, then share information in regard to X_2, work and receive a score.

LEMMA 8. *Let $S_w(x, \hat{x}) = f(x') + 2cx'\hat{x} - c\hat{x}^2$ satisfy best effort and always non-negative. Then for a task with two subtasks, it is always optimal for the worker to split independent subtasks into separate prediction targets.*

PROOF. For any distribution of effort X the worker's expected score from truthful reporting (which is optimal) is

$$E[S_w(X, E[X])] = E[f(X)] + cE[X]^2.$$

To deal with $E[f(X)]$, we make use of two bounds regarding $f(x)$. First, we know that $f'(x') < -2c\hat{x}$ for all $\hat{x}$, so in particular this is true for $\hat{x} = x_{\max}$. By always non-negative, $f(x_{\max}) \geq 0$. Thus, $f(x) \geq (x_{\max} - x)2cx_{\max}$. Second, for $a < b$, $f(a) - f(b) \geq (b - a)2cx_{\max}$. We now show that $E[S_w(X_1, E[X_1])] + E[S_w(X_2, E[X_2])] > E[S_w(X_1 + X_2, E[X_1 + X_2])]$. Note that we use the unconditional expectation over X_2 here because X_1 and X_2 are independent.

$$\begin{aligned}
&E[S_w(X_1, E[X_1])] + E[S_w(X_2, E[X_2])] \\
&- E[S_w(X_1 + X_2, E[X_1 + X_2])] \\
&= E[f(X_1)] + cE[X_1]^2 + E[f(X_2)] + cE[X_2]^2 \\
&- E[f(X_1 + X_2)] - cE[X_1 + X_2]^2 \\
&= E[f(X_1) + f(X_2) - f(X_1 + X_2)] - 2cE[X_1]E[X_2] \\
&\geq E[(x_{\max} - X_1)2cx_{\max} + ((X_1 + X_2) - X_2)2cx_{\max}] \\
&- 2cE[X_1]E[X_2] = 2c(x_{\max}^2 - E[X_1]E[X_2]) > 0.
\end{aligned}$$

□

We take this as a negative observation, because there is no learning effect when splitting out independent subtasks—it is not the case that additional accuracy can be achieved through separate predictions in the absence of correlations.

On the other hand, if we are willing to accept a scoring rule that may be negative, it is easy to obtain a different result. For example, take $f(x') = -kx'$ ($k > 2x_{\max}$) and $c = 1$. Some algebra shows that not splitting results in an increase in utility of $2E[X_1]E[X_2] > 0$, and so independent subtasks are not split out as separate prediction targets.

For this reason, the following is a very helpful observation. If the distinction between the completion of a task and the completion of a subtask is observable by the company, then the scoring system can provide a large enough *bonus score* $B > 0$ upon the completion of a task (but not a subtask), in order to remove the broader implications of a stream of negative scores. We adopt this approach going forward, allowing for scoring rules that may be negative but correcting for this with a large enough catch-up bonus B on the completion of a complete task.[4]

Parameter B can be calculated as the negation of the lowest possible score (the most negative score) that a worker who exerts best effort can possibly get for completing the task. For a given chunk of work (a set of subtasks chosen as a prediction target), the lowest score is achieved when the time to complete it under best effort is maximized while the prediction of the completion time is minimized.

4.2 Correlated Subtasks

To gain a qualitative understanding of the effect of our scoring rules on the propensity to split-out subtasks as separate targets, we adopt a simple model of correlation. The joint distribution on (X_1, X_2) is parameterized with $q \in (1/2, 1]$ and $r \in [0, 1]$. The distribution on time to complete task 1 under best effort is a with probability q and b with probability $1 - q$. With probability r, the time to complete task 2 is the same as for task 1 (i.e. $X_2 = X_1$). Otherwise, with probability $1 - r$ the time to complete task 2 is independently sampled according to probability q.

We use the scoring rule $S_w(x, \hat{x}) = f(x') + 2cx'\hat{x} - c\hat{x}^2$ with $f(x') = C - kx'$ and $c = 1$, where $k > 2x_{\max}$ and C is

[3]In contrast, suppose the score assigned for the completion of a task is negative. In this case, a worker may prefer to spend 10 hours and earn a score of -2 than to spend 1 hour and earn a score of -1, because in those additional 9 hours the worker would be completing additional tasks for more negative scores.

[4]This bonus is invariant to any aspect of the prediction or effort and does not change the rest of the analysis.

a constant. We show that, for appropriate choice of C, the incentive to split-out subtasks increases as r increases, and thus as there is more positive correlation between the time to complete the subtasks under best effort.

In particular, the choice of C sets a threshold for r. If r is below this threshold then the subtasks are independent enough that the worker does not want to split them. If r is above this threshold then the substasks are correlated enough that splitting them to learn is worthwhile. Increasing C decreases this threshold, but increases the cost to the scoring rule. Thus the choice of C allows a trade-off between encouraging the accurate sharing of predictions on subtasks and cost. However, past a certain point, the worker will want to split-out all subtasks regardless, and increasing C will simply increase the cost.

Lemma 9. *Consider a task with two sub-tasks. Let S_w be as above with $C < 2E[X_1]E[X_2]$. Let $r^* = \sqrt{\frac{2E[X_1]E[X_2]-C}{q(1-q)(a-b)^2}}$. If $r \geq r*$ then it is optimal for the worker to split-out subtasks. If $r \leq r^*$ then it is optimal for the worker to not do so.*

Proof. Unlike in Lemma 8, X_1 and X_2 are no longer independent. In particular, this means that the expected score for task two if they are split is no longer simply $E[S_w(X_2, E[X_2])]$. Instead, the worker learns something after completing the first task so, a priori, the expected score is $E_{X_1}[E[S_w(X_2, E[X_2|X_1 = x])|X_1 = x]]$. Hence we can write the expected gain from splitting as follows:

$$
\begin{aligned}
&E[S_w(X_1, E[X_1])] + E_{X_1}[E[S_w(X_2, E[X_2|X_1 = x])|X_1 = x]]\\
&- E[S_w(X_1 + X_2, E[X_1 + X_2])]\\
&= E[C - kX_1 + E[X_1]^2]\\
&+ E_{X_1}[E[C - kX_2 + E_{X_1}[(E[X_2|X_1 = x])^2]|X_1 = x]]\\
&- E[C - k(X_1 + X_2) + E[X_1 + X_2]^2]\\
&= C + E[X_1]^2 + E_{X_1}[(E[X_2|X_1 = x])^2] - E[X_1 + X_2]^2\\
&= C + E[X_1]^2 + E[X_2]^2 - E[X_1 + X_2]^2\\
&+ E_{X_1}[(E[X_2|X_1 = x])^2] - E[X_2]^2\\
&= C - 2E[X_1]E[X_2] + E_{X_1}[(E[X_2|X_1 = x])^2] - E[X_2]^2.
\end{aligned}
$$

For the particularly simple distribution we have chosen, we can expand the last two terms as

$$
\begin{aligned}
&E_{X_1}[(E[X_2|X_1 = x])^2] - E[X_2]^2\\
&= q(ra + (1-r)E[X_2])^2 + (1-q)(rb + (1-r)E[X_2])^2 - E[X_2]^2\\
&= (1-r)^2E[X_2]^2 + qr^2a^2 + (1-q)r^2b^2 + 2qar(1-r)E[X_2]\\
&+ 2(1-q)br(1-r)E[X_2] - E[X_2]^2\\
&= (1-r)^2E[X_2]^2 + qr^2a^2 + (1-q)r^2b^2 + 2r(1-r)E[X_2]^2\\
&- E[X_2]^2\\
&= qr^2a^2 + (1-q)r^2b^2 - r^2E[X_2]^2\\
&= r^2(qa^2 + (1-q)b^2 - (qa + (1-q)b)^2)\\
&= r^2(qa^2 + (1-q)b^2 - q^2a^2 - (1-q)^2b^2 - 2q(1-q)ab)\\
&= r^2((1-q)qa^2 + q(1-q)b^2 - 2q(1-q)ab)\\
&= r^2q(1-q)(a-b)^2
\end{aligned}
$$

Thus, splitting is optimal if and only if $C - 2E[X_1]E[X_2] + r^2q(1-q)(a-b)^2 \geq 0$. Solving for r yields the desired inequalities. □

In a situation where it is important that the cumulative score for each task is non-negative, then a mitigating aspect of this trade-off is that for smaller values of C the scoring system must assign a larger bonus B upon task completion to correct for the possibility of an accumulation of negative scores on subtasks.

Remark: One might also wonder whether it is possible to modify our scoring rules to allow a worker-manager team to "push" new predictions in regard to a particular prediction target over time, and without leading to new strategic considerations. For example, suppose the worker elects not to split-off any subtasks and have as the target the entire task. But now as work is completed on each subtask, perhaps the worker has updated information in regard to when the task will likely be completed. Perhaps surprisingly, the effect of allowing this turns out to be quite subtle.

For example, associating the score with the average of the score from m predictions fails, because the worker-manager team could maximize its realized score by simply pushing a lot of predictions just before completing a task when there is high confidence about how long the task will take. Insisting that predictions are made at fixed intervals of time could lead to a preference for slacking in order to be able to make an additional prediction. Adopting a time-averaged score, integrated over the different predictions made over time in regard to a prediction target, could lead to a preference to work more slowly on subtasks about which the prediction is higher quality. We leave a full reconciliation of this problem to future work.

5. SIMULATIONS

Our simulation study is designed to validate the three qualitative observations in our theoretical analysis: (a) for subtasks with more correlation the worker will tend to split out more subtasks as targets, (b) for a higher value of C the worker will tend to split out more subtasks into targets, and (c) for a higher value of C the average score received by the worker will tend to increase.

For this purpose, we consider a task X with 3 subtasks X_1, X_2, and X_3. With probability q the task is *low* difficulty, and with probability $1-q$ the task is *high* difficulty. Given that the task is low difficulty, then a subtask takes time $a = 0.5$ under best effort with probability $p \in [0.5, 1]$, and $b = 1$ under best effort otherwise. For a high difficulty task, a subtask takes time $b = 1$ with probability p, and $a = 0.5$ otherwise (both under best effort.) In this way, p controls the correlation between effort on subtasks. High p yields high correlation.

We simulate each possible policy a worker might adopt in deciding which subtasks to split-off into separate prediction targets. Altogether, there are six possible policies:

1. Policy 1: Work on each subtask separately. First target is subtask X_1, then X_2, followed by X_3.

2. Policy 2: First target is X_1. If completion time of X_1 is observed to be a then the second target is X_2, followed by X_3. If completion time of X_1 is b then the second target is $X_2 + X_3$ as a chunk.

3. Policy 3: First target is X_1. If completion time of X_1 is observed to be b then the second target is X_2, followed by X_3. If completion time of X_1 is a then the second target is $X_2 + X_3$ as a chunk.

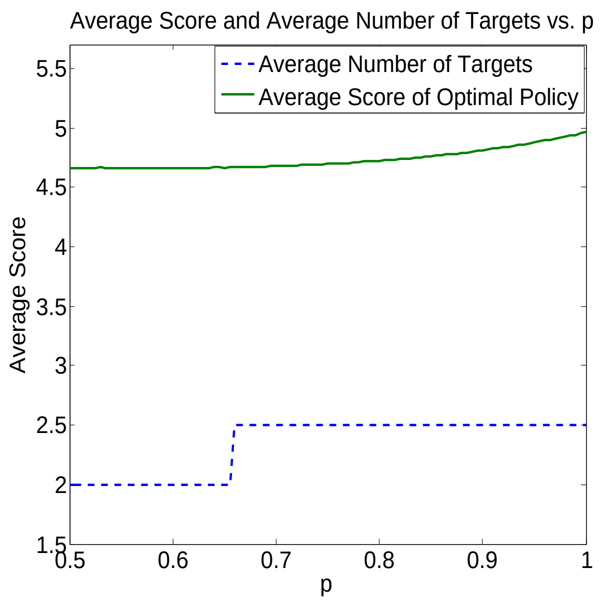


Figure 2: Average score and average number of prediction targets under the best policy, varying $p \in [0.5, 1]$ for $C = -1.9$ and $q = 0.5$.

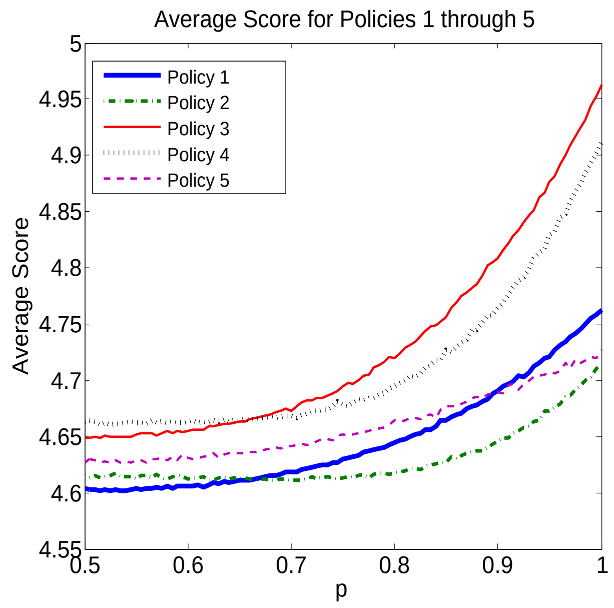


Figure 3: Average score for policies 1 through 5, varying $p \in [0.5, 1]$ for $C = -1.9$ and $q = 0.5$.

4. Policy 4: First target is X_1. The second target is $X_2 + X_3$ as a chunk.

5. Policy 5: First target is $X_1 + X_2$ as a chunk. The second target is X_3.

6. Policy 6: The first and only target is the entire task $X = X_1 + X_2 + X_3$ as a single chunk.

For concreteness, the scoring rule that we adopt is

$$S_w(x, \hat{x}) = C - 2x'x_{\max} + 2x'\hat{x} - \hat{x}^2$$

In considering the score, we also allocate a bonus B upon completion of the entire task, set to the minimal value such that the score is guaranteed to be positive for all contingencies. To determine this value, we first compute all the possible different scores that could be obtained for each policy, selecting the lowest score as that policy's worst score. The (negated) lowest score amongst the 6 worst scores of the 6 policies provides the bonus.

Given this setup, we compare the average score and the average number of prediction targets as the amount of positive correlation (reflected by p) and the parameter in the scoring rule C varies. For each policy, and for different values of C, p and q, we run at least 10,000 trials and determine the average score. The policy that we assume the worker adopts for a triple (C, p, q) is that which maximizes the average score.

Figure 2 is obtained by varying $p \in [0.5, 1]$ for $C = -1.9$ and $q = 0.5$, and shows for each value of p the average score and the average number of targets for the optimal policy *for that value of* p. As p increases there is greater correlation which results in more splitting and a higher score. Figure 3 corroborates this by showing that as p approaches a value of 0.7, the optimal policy changes from Policy 4 to Policy 3. Since Policy 3 varies between 2 and 3 splits, we get an average number of targets equal to 2.5. We have omitted Policy 6 in Figure 3; its average score remained in the range $[2.8, 2.9]$ throughout.

Figure 4 is obtained by varying $C \in [-5, 1]$ for $p = 0.8$ and $q = 0.5$, and shows for each value of C the average score and the average number of targets for the optimal policy *for that value of* C. It shows that as C increases there is more splitting. while the average score also increases. Figure 5 shows the average score of the different policies. The optimal policy is initially Policy 6 (no splitting), and hence there is only 1 prediction target. With increasing C the optimal policy changes to those with greater splitting, finally ending up at Policy 1 (full splitting). We have omitted policies 2 and 5 in Figure 5, as their scores were very close to the scores of policies 3 and 4 respectively (policies 2 and 5 scored slightly less than policies 3 and 4 respectively for all values of C). For $C < -3.75$, the value of B was determined by policy 1, which is why the curve for policy 1 is initially flat and the others are decreasing. For larger values of C, B is determined by policy 6 so it is flat while the others increase.

The basic trends we see in these plots are consistent with the theory, which allows for a tradeoff between the degree to which tasks are split and the cost to the mechanism.

6. CONCLUSION

We have introduced the problem of incentivizing a worker-manager team to commit best effort to a task and make accurate predictions in regard to completion time. In studying this question, we have characterized a family of scoring rules with natural properties, and considered the effect of the rules on decisions in regard to which subtasks to split-out into explicit prediction targets.

The problem was motivated by an extant outcomes-based incentive system currently applied to IBM's internal IT initiatives. In this system, software professionals (developers, software designers, testers, etc.) execute tasks assigned by their project managers to produce project deliverables. Each

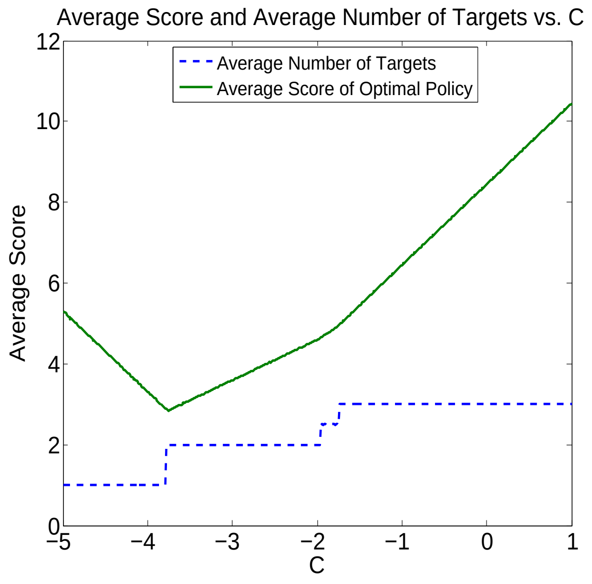


Figure 4: Average score and average number of prediction targets under the best policy, varying $C \in [-5, 1]$ for $p = 0.8$ and $q = 0.5$.

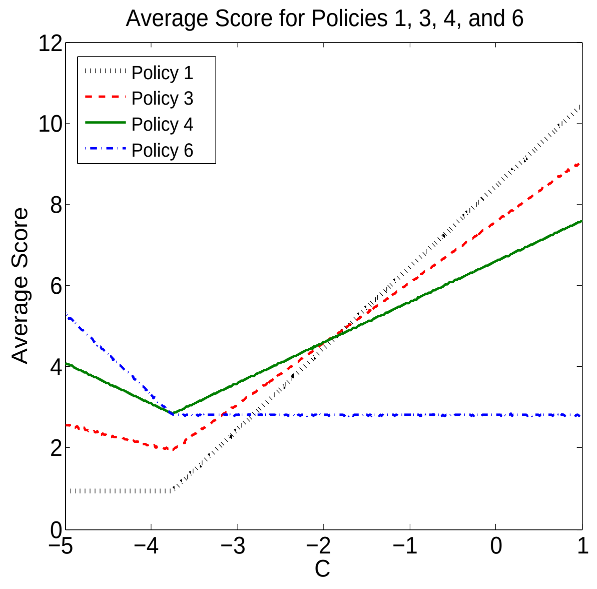


Figure 5: Average score for policies 1, 3, 4, and 6, varying $C \in [-5, 1]$ for $p = 0.8$ and $q = 0.5$.

task is associated with a "Blue Sheet" that records the manager's prediction of required effort for the task, along with its actual completion time. Blue Sheet data are used to compute scores for both 'workers' and 'managers,' and top scorers are recognized for their achievement.

The Blue Sheet system has been in place since 2009 and has provided some useful initial insights on process differences across internal groups. However, the current Blue Sheet scoring system does not satisfy any of the four properties outlined in Section 2.1. It is difficult to derive any strong conclusions about the impact of these missing properties from existing Blue Sheet data (much of the information is self-reported), but the data suggests some evidence of collusion between 'workers' and 'managers.'

We are aiming to pilot a new scoring system based on the current work, comparing to the existing system, both by comparing scores and outcomes and by surveying the participants regarding which system they prefer. It will be interesting to consider, as a next step, additional factors that might be important in a practical deployment. These factors include the impact of a scoring system on the kinds of tasks that worker-manager teams choose to take on, for instance in regard to their inherent predictiveness.

The current Blue Sheet system includes some additional aspects that are outside of our model. These include a self-assessment of the deliverable quality against specified standards, and also an assessment of the extent that re-use of pre-existing assets was leveraged to complete the deliverable. From this perspective, we are interested to understand the impact of a scoring system on how to decompose work into subtasks in the first place, that is on the modularization of tasks. A key goal of the Blue Sheet system is to incentivize the creation and application of reusable software components, thereby making the development process more efficient. Devising incentive schemes that directly encourage creating reusable components, e.g., by rewarding the component author when others reuse the component, remains as future work.

7. ACKNOWLEDGMENTS

We would like to thank Eric Bokelberg of IBM GBS for help with accessing and interpreting Blue Sheet data. This material is based upon work supported by NSF Grant No. CCF-0953516. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors alone.

8. REFERENCES

[1] D. F. Bacon, E. Bokelberg, Y. Chen, I. A. Kash, D. C. Parkes, M. Rao, and M. Sridharan. Software Economies. In *Proc. FSE/SDP Workshop on the Future of Software Engineering Research*, 2010.
[2] D. Fudenberg, and J. Tirole. *Game Theory*. MIT Press, 1991.
[3] T. Gneiting, and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *J. Am. Stat. Assoc.*, 102(477):359–378, March 2007.
[4] R. D. Hanson. Logarithmic market scoring rules for modular combinatorial information aggregation. *Journal of Prediction Markets*, 1(1):1–15, 2007.
[5] N. Lambert, D. M. Pennock, and Y. Shoham. Eliciting properties of probability distributions. In *Proc. 9th ACM Conf. on Electronic commerce* (EC '08), pp. 129–138, 2008.
[6] A. MasColel, M. D. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, 1995.
[7] A. H. Murphy and R. L. Winkler. Probability forecasting in meteorology. *J. Am. Stat. Assoc.*, 79(387):489–500, 1984.
[8] R. Radner. Monitoring Cooperative Agreements in a Repeated Principal-Agent Relationship. *Econometrica*, Vol. 49, No. 5, pp. 1127– 1148, September 1981.
[9] R. Radner. Repeated Principal-Agent Games with Discounting. *Econometrica*, Vol. 53, No. 5, pp. 1173–1198, September 1985.
[10] L. J. Savage. Elicitation of personal probabilities and expectations. *J. Am. Stat. Assoc.*, 66(336):783–801, 1971.
[11] P. Shi, V. Conitzer, and M. Guo. Prediction mechanisms that do not incentivize undesirable actions. In *Proc. 5th International Workshop on Internet and Network Economics* (WINE '09), pp. 89–100, 2009. Springer-Verlag.

Optimal Incentive Timing Strategies for Product Marketing on Social Networks

Pankaj Dayama
Global General Motors R&D - India Science Lab, GM Technical Center,
Bangalore 560066, India
pankaj.dayama@gm.com

Aditya Karnik
Global General Motors R&D - India Science Lab, GM Technical Center,
Bangalore 560066, India
aditya.karnik@gm.com

Y. Narahari
Department of Computer Science and Automation,
Indian Institute of Science,
Bangalore 560012, India
hari@csa.iisc.ernet.in

ABSTRACT

We consider the problem of devising incentive strategies for viral marketing of a product. In particular, we assume that the seller can influence penetration of the product by offering two incentive programs: a) direct incentives to potential buyers (*influence*) and b) referral rewards for customers who influence potential buyers to make the purchase (*exploit connections*). The problem is to determine the optimal timing of these programs over a finite time horizon. In contrast to algorithmic perspective popular in the literature, we take a mean-field approach and formulate the problem as a continuous-time deterministic optimal control problem. We show that the optimal strategy for the seller has a simple structure and can take both forms, namely, *influence-and-exploit* and *exploit-and-influence*. We also show that in some cases it may optimal for the seller to deploy incentive programs mostly for low degree nodes. We support our theoretical results through numerical studies and provide practical insights by analyzing various scenarios.

Categories and Subject Descriptors

J.4 [**Social and Behavioral Sciences**]: Economics

General Terms

Economics, Theory, Performance

Keywords

Incentive Strategies, Social Networks, Viral Marketing

1. INTRODUCTION

A key research topic in multi-agent systems is to understand the effect of microdynamics/interactions between agents on macroscopic properties. Often the agents are a part of a social structure such as a social network. A common example is that of a social network that consists of potential buyers of a particular new product offering in the market. These buyers interact with each other and influence each others' purchase decisions through word-of-mouth and/or behavior. This so-called *social influence* exerted by agents on their neighboring agents in the network have a significant role to play in generating a network effect on the sales of the product. The idea of viral marketing is to essentially exploit the (macroscopic) network effects that result due to the microdynamics between the agents in the network.

Viral marketing is receiving much attention by practicing marketers and academics alike. While not a new idea, it has come to the forefront because of multiple effects - products have become more complex, making buyers to increasingly rely on opinions of their peers; consumers have evolved to distrust advertising; and Web2.0 has revolutionized the way people can connect, communicate and share. With power shifting to consumers, it has become important for sellers to devise effective viral marketing strategies (Godes et al., 2005). This work is motivated by this urgent need.

For social influence to work, there must be *seeds*, i.e., product advocates to start with. The sellers, therefore, employ two basic strategies. The first is to create advocates, by providing incentives to potential buyers to make an actual purchase. These incentives are typically in the form of discounts, free goodies, etc. The second is to reward product advocates who 'put in a good word' and influence potential buyers to make the purchase. Thus, the latter program helps to *exploit* the impact of social influence while making a purchasing decision whereas the former program helps to directly *influence* the buying behavior by offering discounts.

Since incentives come at a cost, a seller must balance the revenue she generates through these strategies and the expenditure she incurs in doing so. This poses some non-trivial challenges. The first is determining incentives themselves, since response of an individual is contingent on them (too low a referral reward may not elicit recommendation from an individual since personal reputation is usually at stake). Secondly, the two programs are not necessarily causally connected. The reputation of a firm or a brand might create product advocates without incentives, thereby, requiring a seller to launch a referral program directly. This necessitates careful 'timing' of these programs.

The objective of this paper is to shed some light on this practically important and theoretically interesting problem. In particular, we seek to determine an optimal timing of these programs over a finite time horizon.

1.1 Related Work

In recent years, problems such as these have attracted much attention. Several papers investigate 'influence maxi-

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

mization' (see, for example, Domingos and Richardson (2001); Kempe, Kleinberg, and Tardos (2003); Bharathi, Kempe, and Salek (2007); Chen, Yaun, and Zhang (2010)), where the problem is to determine the set of initial adopters who, through an influence process, can maximize the future adoptions of the product. Auriol and Benaim (2000) discuss a dynamic model of how standards and norms emerge in decentralized economies. Hartline, Mirrokni, and Sundararajan (2008); Arthur, Motwani, Sharma, and Xu (2009) consider the problem of 'revenue maximization' for viral marketing and are close in spirit to the problem we consider in this paper.

In Hartline et al. (2008) a model is proposed in which the purchase decision of a buyer is influenced by individuals who own the product and the price at which the product is offered. An optimal pricing policy is derived using dynamic programming in a symmetric setting (i.e., identical buyers). In a general setting, finding an optimal strategy is shown to be NP-hard and approximation algorithms are considered. The authors suggest *influence-and-exploit* strategy where selected buyers are given the product for free, and the seller extracts revenue by making a random sequence of offers and a greedy pricing strategy for the remaining buyers to compensate for the initial loss.

Arthur et al. (2009) also considers a model in which a buyer's decision is influenced by friends who own the product and price at which the product is offered. Sales are assumed to cascade through the social network. The seller offers cashback to recommenders and also sets price for each buyer. The authors show that determining an optimal strategy to maximize expected revenue is NP-hard and propose a non-adaptive *influence-and-exploit* policy, which offers product to the interior nodes of the max-leaf spanning tree of the network for free and later exploits their influence by extracting more revenue from the leaf nodes of the tree. They show that the expected revenue generated from the non-adaptive strategy is within a constant factor of the optimal revenue from an adaptive strategy.

1.2 Our Contributions

We consider a seller interested in selling a product to a population of N agents. The product is assumed to be durable and free from network externalities. From the seller's perspective, each agent assumes one of the following types at any point in time: *potential buyer* (one who is yet to make a purchase), *customer* (one who has purchased the seller's product) and *competitor's customer* (one who has purchased a competing product[1]). A potential buyer makes a purchase decision of her own volition (essentially under *external influence*) or under social influence. This decision-making is modeled probabilistically, by specifying for both products (the seller's and the competitor's), probabilities of purchase under external influence and social influence. The seller can influence the former through direct incentives (which affect the price) and the latter through referral rewards. The problem she faces is to roll out these programs so as to maximize the profit, which is equal to the revenue obtained by customer acquisition minus the expenditure on direct incentives and referral rewards, over a given time horizon T.

In practice sellers have limited knowledge about the social network underlying the population, typically, in the form of a *class*-level statistical description of it. A class comprises agents who are considered essentially identical on a variety of factors (chosen by the seller), such as demographic, economic level, number of social contacts and so on. In this paper we consider this set-up. However to keep it simple, we assume heterogeneity only in terms of network connections (in particular, probabilities of purchase under either external or social influence are assumed to be the same for all agents); hence classes are based only on the number of social contacts (degrees). The seller thus knows only the degree distribution and degree-degree correlation of the social network.

This class-level statistical description of the agent population allows us to approximate the stochastic evolution of the purchase dynamics by a deterministic process described by ordinary differential equations (ODEs). This is formally established as a mean-field limit, taking the number of agents $N \to \infty$ Benaim and Boudec (2008). With an ODE limit, we pose the problem as a continuous time optimal control problem and employ the well known Pontryagin's Maximum principle Kirk (1970) to characterize an optimal control. An optimal control specifies for each class the times at which direct incentives and referral rewards programs are to be executed. The following are our main results.

1. We show that an optimal control has a simple structure: the seller needs to run each of the programs at most twice for a certain duration. Moreover, it is non-adaptive (or open-loop). This simplifies the implementation and practically can help a seller pre-allocate the budget for her campaign.

2. While *influence-and-exploit* strategy turns out to be optimal when social influence is strong in the population, *exploit-and-influence* strategy can be optimal when the seller has a good reputation.

3. In some settings, the seller may be better off incentivizing low degree nodes as against the popular approach of targeting the influentials (high degree nodes). This, we believe, provides some support to the findings reported in Watts and Dodds (2007) in reference to the *influentials hypothesis*.

The approach we have taken to address the problem is entirely different from the ones in the literature. While a large size of the population presents a challenge to the earlier approaches, it, in fact, aids us in migrating to a simpler deterministic description of the dynamics. The assumption that agents of a class are indistinguishable also fits in naturally with the popular marketing approach of *customer segmentation* and allows a seller to customize incentives and referral rewards as per these segments.

In contrast to earlier papers, we have also modeled competition. This is not only close to reality but interestingly it allows to address some problems in completely different contexts. For example, in limiting the spread of misinformation about an entity or an Internet virus, the objective is to maximize nodes with correct information or security patches by immunizing them (akin to direct incentives) and/or incentivizing them to spread the information they have to their neighbors (akin to referrals). Our results are, thus, applicable to these problems as well (see Budak et al. (2011) for discussion of the influence limitation problem).

[1]All competitors are aggregated into one single virtual competitor.

2. PROBLEM FORMULATION

Consider a population of N agents, indexed by $i = 1, 2, \ldots, N$. The underlying social network is specified by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each agent is identified with a node in $\mathcal{V}$ and $(i, j) \in \mathcal{E}$ means that i and j are social contacts and they influence each other in decision-making.

s_i denotes the state of agent i. s_i can take three values: 0 (indicates potential buyer), 1 (indicates customer) and -1 (indicates competitor's customer). Let $\mathbf{s} := (s_1, s_2, \ldots, s_N)$.

Each agent makes the purchase decision at a random time point, independent of all others. It suffices to assume that time is discrete (denoted by $n = 1, 2, \ldots$) and at each time step, an agent is chosen uniformly randomly from the population for a potential state change. Since there are no repeat purchases, 1 and -1 are absorbing states. Therefore, the state change occurs only if the chosen agent is a potential buyer. Suppose agent i is chosen at a time n. Then one of the following happens if i is a potential buyer.

1. i buys the seller's product on her own with probability α (For example, $\alpha = 0.08$ means that there is 8% chance that a potential buyer will buy the seller's product on her own).

2. i buys the competitor's product on her own with probability δ.

3. i selects one of her social contacts at random. If the selected contact is a customer, i buys the seller's product under social influence with probability β (For example, $\beta = 0.1$ means that there is 10% chance that a potential buyer will buy the seller's product if she interacts with someone who has already bought the product).

4. i selects one of her social contacts at random. If the selected contact is a competitor's customer, i buys the competitor's product under social influence with probability γ.

Clearly, the state process $\{\mathbf{s}(n), n \geq 1\}$ is a Markov chain.

Now from the seller's perspective, agents having the same degree are indistinguishable and the network $\mathcal{G}$ is known only statistically, i.e., $\mathcal{G}$ is drawn from an ensemble of random undirected graphs of size N, a given degree distribution $P(k)$ $(1 \leq k \leq K)$ and degree-degree correlation function $P(k'|k)$, which denotes the probability that a given link from a node of degree k is to a node of degree k'. Note that a number of well-known graphs such as homogeneous random graphs, exponential random graphs (e.g., $G(n,p)$ and Watts-Strogatz network), scale-free networks can be represented in this framework. We assume that K remains uniformly bounded as $N \to \infty$.

Denote by i_k, r_k and θ_k the fraction of degree-k agents who are potential buyers, customers and competitor's customers respectively (note that normalization is with respect to the number of class-k agents; hence $i_k + r_k + \theta_k = 1$). Let $x_k := (i_k, r_k, \theta_k)$ and $\mathbf{x} := (x_1, \ldots, x_K)$. From the above assumption, it follows that $\{\mathbf{x}(n), n \geq 1\}$ is a Markov chain (as seen by the seller).

The drift of $\mathbf{x}$ can be computed considering the four cases described above. Table 1 shows the corresponding probabilities and the change in x_k for degree class-k. Consider as an example Case 3. The probability of a randomly selected agent being a potential customer of degree k is $P(k)i_k$. This agent randomly chooses one of her k social contacts. The probability that this chosen one is an existing customer is $\frac{1}{k}\sum_{j=1}^{k}\sum_{k'\in K} P(k'|k)r_{k'} = \sum_{k'\in K} P(k'|k)r_{k'}$. The selected agent buys the seller's product under the social influence from her contact with probability β. Thus the probability of Case 3 is $\beta P(k)i_k \sum_{k'\in K} P(k'|k)r_{k'}$. One agent changes her state from 0 (potential customer) to 1 (customer). Hence the effect on x_k is $\frac{1}{NP_k}(-1, 1, 0)$.

We now make the dependence on the population size N explicit and denote by $F^N(\mathbf{x}) := [F_k^N(\mathbf{x})]_{k=1}^K$ the drift of $\mathbf{x}$. $F_k^N(\mathbf{x})$ is as follows.

$$\frac{1}{N}\begin{pmatrix} -\beta i_k \sum_{k'\in K} P(k'|k)r_{k'} - \gamma i_k \sum_{k'\in K} P(k'|k)\theta_{k'} - (\alpha+\delta)i_k \\ \beta i_k \sum_{k'\in K} P(k'|k)r_{k'} + \alpha i_k \\ \gamma i_k \sum_{k'\in K} P(k'|k)\theta_{k'} + \delta i_k \end{pmatrix}$$

Observe that (i) the number of transitions per agent per time slot is of the order of $\frac{1}{N}$ (ii) the second moment of number of agent transitions per time slot is bounded and (iii) $F^N(\mathbf{x})$ is a smooth function of $\frac{1}{N}$ and $\mathbf{x}$. Let $F(\mathbf{x}) = \lim_{N\to\infty} \frac{F^N(\mathbf{x})}{1/N}$. It then follows from Theorem 1 of Benaim and Boudec (2008) that the time evolution of $\mathbf{x}(n)$ can be approximated by the following system of ODEs (with the same initial conditions).

$$\dot{\mathbf{x}} = F(\mathbf{x}) \tag{1}$$

More explicitly, for $1 \leq k \leq K$

$$\begin{aligned} \dot{i}_k &= -\beta i_k R_k - \gamma i_k \Theta_k - (\alpha + \delta) i_k \\ \dot{r}_k &= \beta i_k R_k + \alpha i_k \\ \dot{\theta}_k &= \gamma i_k \Theta_k + \delta i_k \end{aligned}$$

where, $R_k := \sum_{k'\in K} P(k'|k)r_{k'}$ and $\Theta_k := \sum_{k'\in K} P(k'|k)\theta_{k'}$.

The seller offers direct incentives and referral rewards to increase α and β respectively. We model this as follows. A referral reward of c results in an increase of ϵ_1 in β and a direct incentive of c' causes α to increase by ϵ_2. Thus, for the duration of the referral reward program, social influence rate of $(\beta + \epsilon_1)$ is operational and the seller incurs a cost of c for every successful referral. Similarly, if the direct incentive program is executed for some duration, the take-rate for seller's product increases to $(\alpha + \epsilon_2)$ for that duration, incurring her a cost of c' for every sale. We normalize c and c' with respect to the product price. Thus the price is fixed to 1. The seller's problem of maximizing her profit (revenue minus cost) over a fixed time horizon T by optimally timing the two program can now be stated formally as follows.

Let $u_k(t)$ (resp. $v_k(t)$) denote the control variable indicating whether or not the referral reward program (resp. direct incentive program) is offered to class-k at time t. The cost

Case	Probability	Effect on x_k
1	$\alpha P(k)i_k$	$\frac{1}{NP_k}(-1,1,0)$
2	$\delta P(k)i_k$	$\frac{1}{NP_k}(-1,0,1)$
3	$\beta P(k)i_k \sum_{k'\in K} P(k'\|k)r_{k'}$	$\frac{1}{NP_k}(-1,1,0)$
4	$\gamma P(k)i_k \sum_{k'\in K} P(k'\|k)\theta_{k'}$	$\frac{1}{NP_k}(-1,0,1)$

Table 1: Probability and effect on x_k for different cases

incurred in running the referral reward program is

$$\int_0^T \sum_{k=1}^K P(k)u_k(t)c(\beta+\epsilon_1)i_k(t)R_k(t)dt \qquad (2)$$

Recall that the *conversion rate* of potential buyers under the program is $c(\beta+\epsilon_1)i_k(t)R_k(t)$. The cost incurred in the direct incentives program is

$$\int_0^T \sum_{k=1}^K P(k)v_k(t)c'(\alpha+\epsilon_2)i_k(t)dt \qquad (3)$$

Since the product price is unity, the revenue obtained is proportional to the number of customers at the end of horizon, $\sum_{k=1}^K P(k)r_k(T)$. Denoting the total cost (2)+(3) by $C(T)$ the problem is

$$\text{Maximize } \sum_{k=1}^K P(k)r_k(T) - C(T)$$

subject to

$$\begin{aligned}\dot{i}_k &= -(\beta+u_k\epsilon_1)i_kR_k - (\alpha+v_k\epsilon_2)i_k - \gamma i_k\Theta_k - \delta i_k\\ \dot{r}_k &= (\beta+u_k\epsilon_1)i_kR_k + (\alpha+v_k\epsilon_2)i_k\\ \dot{\theta}_k &= \gamma i_k\Theta_k + \delta i_k\end{aligned}$$

for $1 \le k \le K$ and a given initial condition $\mathbf{x}(0)$.

Three remarks are in order. The assumption of heterogeneity only in the number of social contacts is mainly to keep the formulation simple and highlight the impact of network structure. Extending this formulation to a general setting is straightforward and will be taken up in a longer version of the paper. Our random interaction model essentially means that the social influence on a potential buyer is the average influence from her neighbors. This, we believe, is reasonable since we have also assumed presence of external influence (through α) on agents[2]. In the above formulation we consider fixed rewards and incentives pay-outs (c and c'). This simplifies implementation in practice. Calibration of c and c' can be carried out through numerical studies.

3. STRUCTURE OF OPTIMAL CONTROL

In this section we mathematically prove the structural properties of an optimal control. To keep the proof simple, we will assume that the network $\mathcal{G}$ is drawn randomly from a set of regular networks of size N and degree k. This is without loss of generality.

Let $i(t)$, $r(t)$ and $\theta(t)$ denote the fraction of population in states $\{0,1,-1\}$ at time t respectively. Let $u(t) \in \{0,1\}$ denote whether or not the referral reward program is offered at time t and let $v(t) \in \{0,1\}$ denote whether or not the direct incentive program is offered at time t. The purchase dynamics under the influence of these programs are given as follows:

$$\begin{aligned}\dot{i} &= -(\beta+u\epsilon_1)ir - (\alpha+v\epsilon_2)i - \gamma i\theta - \delta i && (4)\\ \dot{r} &= (\beta+u\epsilon_1)ir + (\alpha+v\epsilon_2)i && (5)\\ \dot{\theta} &= \gamma i\theta + \delta i && (6)\end{aligned}$$

From (4), (5), and (6), observe that $\dot{i}+\dot{r}+\dot{\theta}=0$. Therefore, it suffices to consider any two equations. Let $\Omega := \{(i,r)|i+r \le 1, i \ge 0, r \ge 0\}$. Let $\mathbf{x}(t) := (i(t),r(t)) \in \Omega$ denote the state variable.

[2]For the lack of clear empirical evidence, one may also consider *total* influence from the neighbors. Mathematically, it is a simple modification to our formulation.

The optimal control problem in this simpler setting is as follows.

$$\begin{aligned}\text{Maximize} \qquad & r(T) - \int_0^T cu(t)(\beta+\epsilon_1)i(t)r(t)dt\\ & - \int_0^T c'v(t)(\alpha+\epsilon_2)i(t))dt \qquad (7)\end{aligned}$$

subject to (4), (5) and the following constraints on state and control variables: for all $0 \le t \le T$, $\mathbf{x}(t) \in \Omega$, $u(t) \in \{0,1\}$ and $v(t) \in \{0,1\}$.

Our main result is given in Proposition 1. It shows that an optimal strategy for the seller is to deploy the two incentive programs for at most two distinct time periods.

PROPOSITION 1. *1. There exist τ_1, τ_2 ($0 \le \tau_1 \le \tau_2 \le T$) such that $u^*(t) = 0$ for $\tau_1 < t \le \tau_2$ and $u^*(t) = 1$ else.*

2. There exist τ_3, τ_4 ($0 \le \tau_3 \le \tau_4 \le T$) such that $v^*(t) = 0$ for $\tau_3 < t \le \tau_4$ and $v^*(t) = 1$ else.

PROOF. $i(0) > 0$ otherwise there is no problem to solve. Observe that Ω is positively invariant. Therefore a solution starting from any initial point $\mathbf{x}(0) \in \Omega$ remains confined to Ω. This allows us to disregard state constraints from the control formulation.

Let $u(t), v(t) \in [0,1]$ for all $t \in [0,T]$ (This relaxation allows us to establish existence of an optimal control. We show that the optimal controls are indeed 'bang-bang', i.e., $u^*(t), v^*(t) \in \{0,1\}$ for all t). Writing the problem in Mayer form, it can be seen that the state space (appropriately expanded with additional variables) is bounded and positively invariant (thus, state trajectories remain bounded for all admissible pairs); and the system is affine in controls (see (7), (4) and (5)). Existence of an optimal control is now established by Filippov-Cesari theorem.

From (4), (5), and (7), the Hamiltonian is written as follows.

$$\begin{aligned}H(\mathbf{x},\mathbf{p},u,v) &= -cu(\beta+\epsilon_1)ir - c'v(\alpha+\epsilon_2)i\\ &\quad -p_1[(\beta+u\epsilon_1)ir + (\alpha+v\epsilon_2)i + \gamma i\theta + \delta i]\\ &\quad +p_2[(\beta+u\epsilon_1)ir + (\alpha+v\epsilon_2)i] \qquad (8)\end{aligned}$$

$\mathbf{p} := (p_1,p_2)$ denotes co-state variables. Then according to Pontryagin's Maximum Principle, there exist continuous and piecewise continuously differentiable co-state functions p_1 and p_2 that satisfy

$$\begin{aligned}\dot{p}_1 &= -\frac{\partial H}{\partial i}\\ &= [c\beta - (p_2-p_1-c)\epsilon_1]ru + [c'\alpha - (p_2-p_1-c')\epsilon_2]v\\ &\quad +(p_1-p_2)(\beta r+\alpha) + p_1(\gamma(1-2i-r)+\delta) \qquad (9)\\ \dot{p}_2 &= -\frac{\partial H}{\partial r}\\ &= [c\beta - (p_2-p_1-c)\epsilon_1)]ui + (p_1-p_2)\beta i - p_1\gamma i, \quad (10)\end{aligned}$$

at all $t \in [0,T]$ where u and v are continuous and satisfy the following transversality condition

$$p_1^*(T) = 0,\ p_2^*(T) = 1. \qquad (11)$$

and also satisfy, for all $t \in [0,T]$, $u(t) \in [0,1]$ and $v(t) \in$

$[0, 1]$,

$$\begin{array}{l} H(x^*(t), p^*(t), u^*(t), v(t)) \geq H(x^*(t), p^*(t), u(t), v(t)) \\ H(x^*(t), p^*(t), u(t), v^*(t)) \geq H(x^*(t), p^*(t), u(t), v(t)). \end{array} \quad (12)$$

From (8) and (12), we get the following form for controls.

$$u^*(t) = \begin{cases} 1 & \text{if } (p_2^*(t) - p_1^*(t) - c)\epsilon_1 > c\beta \\ 0 & \text{if } (p_2^*(t) - p_1^*(t) - c)\epsilon_1 < c\beta \end{cases} \quad (13)$$

$$v^*(t) = \begin{cases} 1 & \text{if } (p_2^*(t) - p_1^*(t) - c')\epsilon_2 > c'\alpha \\ 0 & \text{if } (p_2^*(t) - p_1^*(t) - c')\epsilon_2 < c'\alpha \end{cases} \quad (14)$$

In case of equality in the conditions specified in equations (13) and (14), $u^*(t)$ and $v^*(t)$ may take any arbitrary values in $[0, 1]$.

Let $\phi(t) := (p_2^*(t) - p_1^*(t) - c)\epsilon_1 - c\beta$ and $\psi(t) := (p_2^*(t) - p_1^*(t) - c')\epsilon_2 - c'\alpha$.

We denote by H_t^* the Hamiltonian along optimal state-control trajectory at time t. The following lemma proves that Hamiltonian will always remain positive.

LEMMA 1. $H_t^* > 0 \; \forall \; t \in [0, T]$.

PROOF. From (8) and (11), we have

$$\begin{aligned} H_T^* &= [(1-c)\epsilon_1 - c\beta] i^*(T) r^*(T) u^*(T) \\ &\quad + [(1-c')\epsilon_2 - c'\alpha] i^*(T) v^*(T) \\ &\quad + (\beta i^*(T) r^*(T) + \alpha i^*(T)) \end{aligned}$$

$r(t)$ is non-decreasing whereas $i(t) \downarrow 0$ and $i(t) > 0$ for all t since $i(0) > 0$. Therefore, $H_T^* > 0$. The conclusion follows by noting that the Hamiltonian is constant for autonomous systems. □

The lemma below shows that the co-state variables remain positive for the whole duration.

LEMMA 2. $p_1^*(t), p_2^*(t) > 0 \; \forall \; t \in [0, T)$.

PROOF. Suppose $p_1^*(t) \leq 0$ for all t and let $p_1^*(t) = 0$ at $t = \tau$ (at least one τ exists since $p_1^*(T) = 0$). Then $p_2^*(\tau) > 0$ otherwise $H_\tau^* < 0$ since $u^*(\tau) = 0$. Observe from (9) that $\dot{p}_1 < 0$ if $p_1 = 0$. Strict inequality in (13) implies that at τ, $u^*(\cdot)$ is continuous. Therefore, $\dot{p}_1 < 0$ in the neighborhood of 0. Thus $p_1^*(t_1) \leq 0$ implies $p_1^*(t) < 0$ for all $t > t_1$ and $p_1^*(T) \neq 0$ which violates (11). It follows that $p_1^*(t) > 0$ for all $t \in [0, T)$. This in turn implies that $p_2^*(t) > 0$ for all t otherwise $H_t^* < 0$. □

LEMMA 3. $p_2^*(t) > p_1^*(t) \; \forall \; t \in [0, T]$.

PROOF. Suppose not. Let $p_2^*(t) < p_1^*(t)$ at $t = \tau$. Then $\phi(\tau) < 0$ and, therefore, $u^*(\tau) = 0$. (8) then yields $H_\tau^* < 0$, a contradiction. □

Let $\zeta(t) := (p_2^*(t) - p_1^*(t)) i^*(t)$. The lemma that follows shows that $\zeta(t)$ is a decreasing function.

LEMMA 4. $\dot{\zeta}(t) < 0 \; \forall \; t \in [0, T]$.

PROOF. From (4), (9), and (10) we get

$$\begin{aligned} \dot{\zeta}(t) &= -[c(\epsilon_1 + \beta) u^*(t) r^*(t) + \phi(t) u^*(t) i^*(t) \\ &\quad + c'(\epsilon_2 + \alpha) v^*(t) \\ &\quad + p_2^*(t)(\gamma(1 - i^*(t) - r^*(t)) + \delta)] i^*(t) \end{aligned}$$

Lemma follows by noting that all terms inside the bracket are non-negative. □

Now consider $\dot{\phi}(t)$. From (9), (10), and (8) we get

$$\dot{\phi}(t) = [\frac{H_t^*}{i^*(t)} - \phi(t) i^*(t) u^*(t) - (p_2^*(t) - p_1^*(t))\beta i^*(t)]\epsilon_1$$

$(p_2^*(t) - p_1^*(t)) i^*(t)$ is monotonically decreasing (Lemma 4). From (4) $i^*(t) \downarrow 0$ exponentially ($i^*(t) < i^*(0) e^{-(\alpha+\delta)t}$). H_t^* is a positive constant (Lemma 1).

Assume that $\phi(t) = 0$ at three points in time τ_1, τ_2, τ_3. Therefore, $\dot{\phi}(\tau) > 0$ for either $\tau = \tau_1$ or $\tau = \tau_2$. Without loss of generality, let us say $\dot{\phi}(\tau_2) > 0$. From the above equation it follows that $\dot{\phi}(\tau_3) > 0$ which is not feasible as $\phi(\tau_3^-) > 0$. It follows that $\phi(t) = 0$ at at most two points in time. Therefore, there exist $0 \leq \tau_1 \leq \tau_2 \leq T$ such that $u^*(t) = 1$ for $0 \leq t \leq \tau_1$ and $\tau_2 < t \leq T$, and 0 elsewhere.

Similarly, one can show that there exist $0 \leq \tau_3 \leq \tau_4 \leq T$ such that $v^*(t) = 0$ for $\tau_3 < t \leq \tau_4$ and $v^*(t) = 1$ otherwise. The proposition is, thus, established. ■

Proposition 1 implies that both the referral reward and direct incentives programs are to be deployed at most twice for certain durations, one in the beginning and the other at the end. It may happen that both the durations are of length 0 which means that a program is not deployed at all. On the other hand, it could also get deployed over the complete time horizon T. This gives a simple and elegant marketing strategy which is easy to implement for the seller.

The structure of the above optimal control is quite intuitive. In the case of the referral reward program, the cost is proportional to the product of number of potential buyers and customers. Hence to keep the cost low, rewards are declared in the initial stage (when the number of customers is less) to motivate product advocates and may also be paid at the end (when the number of potential buyers is less) to acquire some additional customers.

In the case of direct incentives, the cost is proportional to the number of potential buyers. If the initial take rate for the product is less, this program may get executed at initial stages to quickly acquire customers whose social influence can be exploited in the later stages; otherwise more agents may buy competitor's product and attract other potential buyers. Towards the end of the campaign, the number of potential buyers is less; hence direct incentives may be offered to attract additional customers.

4. NUMERICAL RESULTS

The simple structure of optimal controls given by Proposition 1 allows one to devise incentives programs quite easily by numerical optimization of τ's. Here we obtain an independent validation of optimal controls by discretizing (7), casting it as a nonlinear constrained optimization problem and using a gradient descent approach to find an optimal solution. (For discussion on various numerical solution techniques for such problems refer to Kirk (1970)). For all our experiments, the time horizon T and discretization step-size are fixed at 10 and 0.1 respectively. The NLP formulation is not convex. Therefore, we use a multi-start mechanism to determine an optimal solution. Results are also verified using the commercial package PROPT which uses pseudospectral methods for solving such problems.

In this paper, we will primarily investigate the initial condition $i(0) = 1$. This captures the case when the seller and the competitor(s) enter the market with substitutable products at around the same time (e.g., gaming technologies)

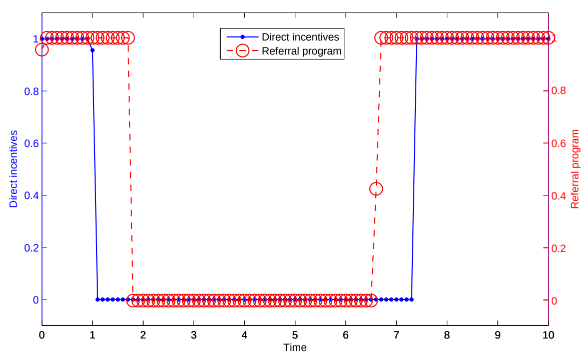


Figure 1: Optimal marketing strategy for the base scenario

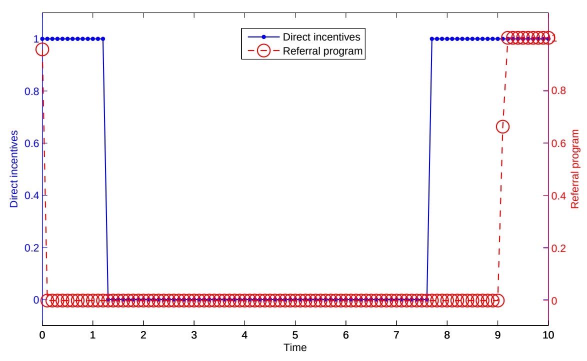


Figure 2: *Influence-and-exploit* strategy is optimal when β is increased to 0.13

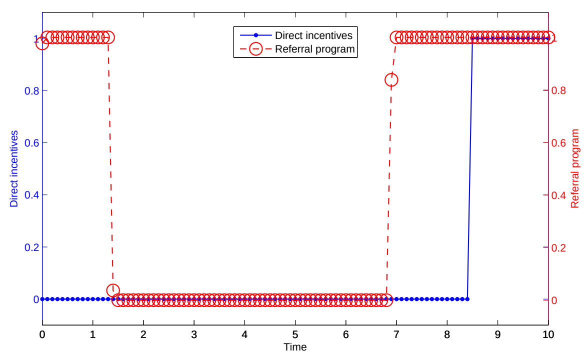


Figure 3: *Exploit-and-influence* strategy is optimal when α is increased to 0.09

or when the seller introduces an independent product into the market (e.g., a book). Of course, similar results can be obtained for the case where seller and/or competitor already have some presence in the market ($r(0) > 0$ and/or $\theta(0) > 0$).

We fix $\epsilon_1 = \epsilon_2 = 0.05, \gamma = 0.1$, $\delta = 0.1$ for all numerical studies, and consider $\beta = 0.1, \alpha = 0.08, c = 0.25$ and $c' = 0.3$ as the *base scenario* (These parameter values are arbitrary and only roughly based on some available data). The optimal marketing strategy is shown in Figure 1. It is optimal for the seller to run both the incentive programs initially for some duration, stop and then run the programs again towards the end. Note that the optimal strategy is open loop; hence estimates of i and r are not required for implementation.

It is, thus, possible for the seller to determine the timing of her incentives programs numerically. Experimentation with different values of pay-outs c and c' (which essentially fix ϵ_1 and ϵ_2) can be used to understand trade-offs and optimize these pay-outs.

In the following we undertake an investigation of two important questions pertaining to the interplay between the two incentives programs and the impact of network structure on the them. The former question is important because *influence-and-exploit* strategy has received much attention in the literature. As we show below, *exploit-and-influence* strategy can also come into play for some parameter settings. The second question is linked to the so-called *influentials hypothesis* which informally says that high degree agents (*hubs*) play significant role in product diffusion, and, therefore, are natural targets for incentives (direct or referral rewards). We show that in some cases the seller is better off incentivizing low degree agents (more than high-degree ones). Thus, our results highlight the need for a careful consideration of the network structure while making incentive decisions.

4.1 Interplay between Referral and Direct Incentive Programs

When social influence is strong in the population, i.e., β is higher, the seller needs to employ only direct incentives initially. For example, if β is set to 0.13 in the base scenario then it is optimal to offer referral rewards only at the end and that too for a short period as shown in Figure 2. This can be seen as a manifestation of the *influence-and-exploit* strategy. On the other hand, if the seller has established a good reputation in the market, translating into a higher value of α, an initial influence step through direct incentives may not even be required. See from Figure 3 that when $\alpha = 0.09$ in the base scenario, it is optimal for the seller to offer direct incentives only at the end. In this case, for most portion of the time horizon, the seller must exploit connections of existing customers and only at the end must she impart direct influence on potential buyers. We call it the *exploit-and-influence* strategy for the seller.

We also observe from Figures 4 and 5 that *influence-and-exploit* and *exploit-and-influence* are optimal strategies for the seller if she incurs high per conversion pay-outs for referral and incentive programs respectively.

4.2 Impact of Network Structure on Incentive Programs

Real-world networks show strong degree correlation amongst connected nodes. Some networks show *assortative* mixing of nodes by degrees where high-degree nodes have most of their connections to other high-degree nodes. Others show *disassortative* mixing where high-degree nodes have most of their connections to low-degree nodes (Newman (2002)). In this section, we examine the impact of network structure on incentives programs.

We consider an undirected correlated network with nodes belonging to either of two classes A and B with probability $P(A)$ and $P(B)$ respectively. Class A nodes are of high degree, say k_A and class B nodes are of low degree, say k_B.

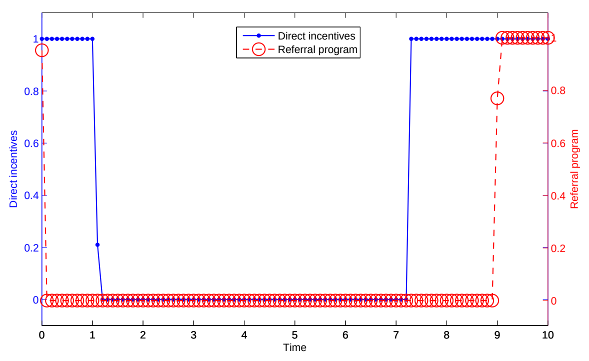


Figure 4: *Influence-and-exploit* strategy is optimal with pay-outs: $c = 0.3$, $c' = 0.3$

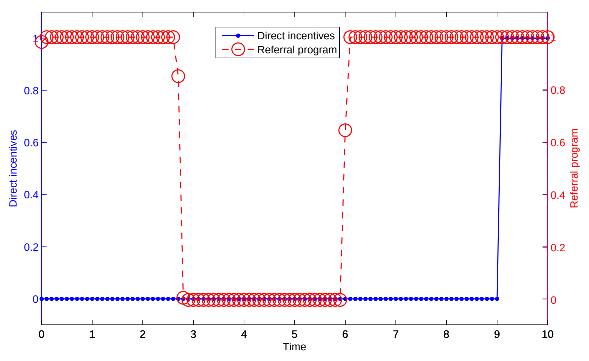


Figure 5: *Exploit-and-influence* strategy is optimal with pay-outs: $c = 0.25$, $c' = 0.35$

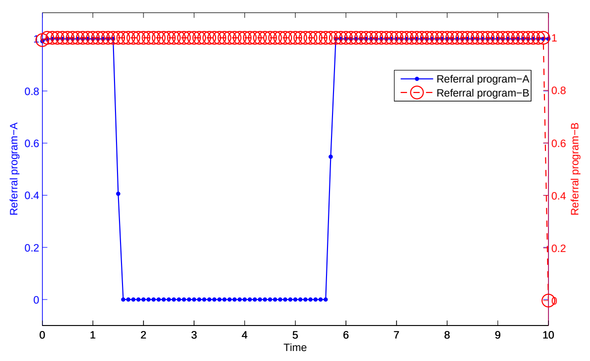


Figure 6: Optimal marketing strategy on a network with disassortative mixing

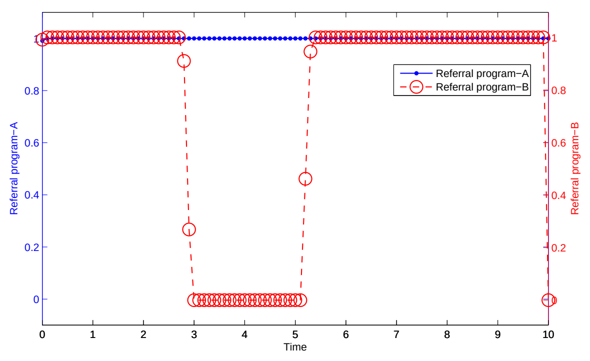


Figure 7: Optimal marketing strategy on a network with assortative mixing

$P(A|B)$ is the probability that a given link from class B node points to a class A node. $P(B|A)$ can be computed from the following balance equation:

$$k_A P(B|A)P(A) = k_B P(A|B)P(B) \tag{15}$$

We consider two types of network structures. One structure represents assortative mixing whereas the other one represents disassortative mixing. To keep things simple and derive key insights, we assume that the seller is optimizing implementation of only one incentives program, namely, referral rewards program. The seller can offer referral rewards to class A and/or class B nodes to increase their social influence rate (β) by ϵ. As earlier, she incurs a per conversion cost of c after normalizing with respect to product price.

We fix $\alpha = 0.1, \delta = 0.1, \beta = 0.1, \gamma = 0.15, \epsilon = 0.08, k_A = 10, k_B = 2, P(A) = 0.1, P(B) = 0.9$ for our numerical studies. For disassortative network, we set $P(B|A) = 0.9$ whereas for assortative network, we set $P(B|A) = 0.1$.

The optimal timing of referral reward program for disassortative network is shown in Figure 6. The seller's optimal strategy is to offer referral rewards to class B nodes for the complete duration whereas rewards to class A nodes are offered initially for a short duration and then again towards the end for a short duration. In this case, class B nodes have almost half of their connections going to class A nodes. Also, major fraction of the population is from class B. So, referral rewards are offered to class B nodes for entire duration as it increases influence not only on class B nodes but also on class A nodes. Class A nodes are not rewarded for the entire duration in order to control the cost.

In the case of assortative network, the optimal strategy changes completely (see Figure 7). The seller offers referral rewards to class A nodes for the complete duration whereas rewards to class B nodes are offered initially for some duration and then again towards the end. In this case, nodes from both the classes are well connected amongst themselves with very few connections going across the classes. So, the optimal reward strategies for both the classes are essentially independent. For this particular scenario, it turns out that it is optimal to offer rewards to class A nodes for the entire duration as the cost incurred is not much. Whereas in the case of class B nodes, referral rewards are discontinued for some duration in the middle as the cost overshoots the potential revenue.

The results show that networks with different structures can result in different optimal strategies for the seller. In some scenarios, the seller may be better off incentivizing low degree nodes as against the popular approach of targeting the influentials (high degree nodes), thus, providing some support to the finding in Watts and Dodds (2007). In some scenarios, the seller may be better off targeting influentials thus supporting the results in Goldenberg et al. (2009). Thus, our results highlight the need for a careful consideration of the network structure while making incentive decisions.

5. CONCLUSION

In this paper we have addressed the problem of optimal timing of two incentive programs, namely, direct incentives and referral rewards, for product diffusion through social networks. Taking a deviation from the existing approaches, we formulate the problem as a continuous-time deterministic optimal control problem. The optimal strategy for the seller is to deploy these programs in at most two distinct time periods. The simplicity of this structure and non-adaptive nature makes them ideal for implementation in practice. We further show that if the seller has good reputation in the market, *exploit-and-influence* strategy can be optimal whereas if social influence is strong in the population, *influence-and-exploit* strategy can be optimal for the seller. In the case of correlated networks, our numerical studies show that the seller need not necessarily offer more frequent referral reward programs to high degree nodes to maximize her profit.

There are two immediate directions for future work: (*i*) extend heterogeneity of agents to include their external and social influence probabilities and (*ii*) devise procedures to estimate model parameters.

References

D. Arthur, R. Motwani, A. Sharma, and Y. Xu. Pricing strategies for viral marketing on Social Networks. In *Proceedings of Workshop on Internet and Network Economics*, pages 101–112, 2009.

E. Auriol and M. Benaim. Standardization in decentralized economies. *American Economic Review*, 90(3):550–570, 2000.

M. Benaim and J. Y. Le Boudec. A Class of Mean Field Interaction Models for Computer and Communication Systems. *Performance Evaluation*, 65:823–838, 2008.

S. Bharathi, D. Kempe, and M. Salek. Competitive influence maximization in social networks. In *Proceedings of Workshop on Internet and Network Economics*, pages 306–311, 2007.

C. Budak, D. Agrawal, and A. El Abbadi. Limiting the Spread of Misinformation in Social Networks. In *Proceedings of the 20th International Conference on World Wide Web*, pages 665–674, 2011.

W. Chen, Y. Yaun, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *Proceedings of 10th International Conference on Data Mining*, 2010.

P. Domingos and M. Richardson. Mining the Network Value of Customers. In *7th International Conference on Knowledge Discovery and Data Mining*, pages 57–66, 2001.

D. Godes, D. Mayzlin, Y. Chen, S. Das, C. Dellarocas, B. Pfeiffer, B. Libai, S. Sen, M. Shi, and P. Verlegh. The Firm's Management of Social Interactions. *Marketing Letters*, 16(3/4):415–428, 2005.

J. Goldenberg, S. Han, D. R. Lehmann, and J. W. Hong. The Role of Hubs in the Adoption Processes. *Journal of Marketing*, 73:1–13, March 2009.

J. Hartline, V. S. Mirrokni, and M. Sundararajan. Optimal Marketing Strategies over Social Networks. In *Proceedings of 17th International Conference on World Wide Web*, pages 189–198, 2008.

D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the Spread of Influence through a Social Network. In *Proceedings of Knowledge Discovery and Data Mining*, pages 137–146, 2003.

D. E. Kirk. *Optimal Control Theory: An Introduction.* Dover, 1970.

M. E. J. Newman. Assortative Mixing in Networks. *Physics Review Letters*, 89(20), November 2002.

D. Watts and P. S. Dodds. Influentials, Networks, and Public Opinion Formation. *Journal of Consumer Research*, 34, December 2007.

Optimizing Kidney Exchange with Transplant Chains: Theory and Reality

John P. Dickerson Ariel D. Procaccia Tuomas Sandholm

Department of Computer Science
Carnegie Mellon University
{dickerson,arielpro,sandholm}@cs.cmu.edu

ABSTRACT

Kidney exchange, where needy patients swap incompatible donors with each other, offers a lifesaving alternative to waiting for an organ from the deceased-donor waiting list. Recently, *chains*—sequences of transplants initiated by an altruistic kidney donor—have shown marked success in practice, yet remain poorly understood. We provide a theoretical analysis of the efficacy of chains in the most widely used kidney exchange model, proving that long chains do not help beyond chains of length of 3 in the large. This completely contradicts our real-world results gathered from the budding nationwide kidney exchange in the United States; there, solution quality improves by increasing the chain length cap to 13 or beyond. We analyze reasons for this gulf between theory and practice, motivated by our experiences running the only nationwide kidney exchange. We augment the standard kidney exchange model to include a variety of real-world features. Experiments in the static setting support the theory and help determine how large is really "in the large". Experiments in the dynamic setting cannot be conducted in the large due to computational limitations, but with up to 460 candidates, a chain cap of 4 was best (in fact, better than 5).

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems; J.4 [**Social and Behavioral Sciences**]: Economics

General Terms

Economics, Theory, Experimentation

Keywords

Kidney exchange

1. INTRODUCTION

The role of kidneys is to filter waste from blood. Kidney failure results in accumulation of this waste, which leads to death in months. One treatment option is dialysis, in which the patient goes to a hospital to have his/her blood filtered by an external machine. Several visits are required per week, and each takes several hours. The quality of life on dialysis can be extremely low, and in fact many patients opt to withdraw from dialysis, leading to a natural death. Only 12% of dialysis patients survive 10 years [18].

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

Instead, the preferred treatment is a kidney transplant. Kidneys are by far the most common organ to transplant—more prevalent than all other organ transplants combined. Unfortunately, the demand for kidneys far outstrips supply. In the United States alone, in 2010, 4,654 people died waiting for a life-saving kidney transplant. During this time, 34,418 people were added to the national waiting list, while only 10,600 people left the list by receiving a deceased-donor kidney. The waiting list has 89,808 people, and the median waiting time is between 2 to 5 years, depending on blood type [16].

For many patients with kidney disease, the best option is to find a *living* donor, that is, a healthy person willing to donate one of his/her two kidneys. Although there are marketplaces for buying and selling living-donor kidneys, the commercialization of human organs is almost universally regarded as unethical, and the practice is explicitly illegal in most countries. However, in most countries, live donation is legal, provided it occurs as a gift with no financial compensation. In 2010, there were 5,467 live donations in the US.

The number of live donations would have been much higher if it were not for the fact that, in most cases, a potential donor and his intended recipient are blood-type or tissue-type incompatible. In the past, the incompatible donor was sent home, leaving the patient to wait for a deceased-donor kidney. This is where kidney exchanges come into play, in which patients can swap their incompatible donors with each other, in order to each obtain a compatible donor. While still in their infancy, kidney exchanges have now been fielded at the regional and national level.

In this paper, we consider altruistic chains, a recent innovation for barter exchanges that has been widely adopted for kidneys, but is poorly understood. Section 2 describes the formal exchange *clearing problem* and why chains exacerbate the already computationally intractable problem. Section 3 reports results from the first (and only) nationwide kidney exchange, using our fielded technology; these real-world results clearly show the benefit of integrating chains into the clearing process. Section 4 formalizes the theoretical benefit of chains as a kidney exchange scales to the large, and Section 5 experimentally determines exactly what "large" means. Section 6 studies the dynamics of kidney exchange over time, using an extension over the state-of-the-art model to more accurately represent the realities of modern kidney exchange.

2. THE CLEARING PROBLEM

One can encode an n-patient kidney exchange (and almost any n-agent barter exchange, such as Netcycler for used goods, Read-It-Swap-It for used books, the National Odd Shoe Exchange, and Intervac for exchanging time in holiday homes) as a directed graph $G(n)$ as follows. Construct one vertex for each patient. Add a weighted edge e from one patient v_i to another v_j, if v_j wants the item of v_i. In the context of kidney exchange, the item is a kidney from a donor that v_i brings with him into the exchange; the donor is willing to give a kidney if and only if v_i receives a kidney. The weight w_e of edge e represents the utility to v_j of obtaining v_i's item. In kidney exchange, the methodology for setting weights is decided by the exchange design committee. The weights take into account such considerations as age, degree of compatibility, wait time, and geographic proximity. A cycle c in this graph represents a possible swap, with each agent in the cycle obtaining the item of the next agent. The weight w_c of a cycle c is the sum of its edge weights. An *exchange* is a collection of disjoint cycles. (They have to be disjoint because no donor can give more than one kidney.)

The vanilla version of the *clearing problem* is to find a maximum-weight exchange consisting of cycles with length at most some small constant L (typically, $2 \leq L \leq 5$). This cycle-length constraint is crucial. For one, all operations in a cycle have to be performed simultaneously; otherwise a donor might back out after his incompatible partner has received a kidney.[1] The availability of operating rooms, doctors, and staff thus constrains cycle length.

The clearing problem with $L > 2$ is NP-complete [1]. Yet significantly better solutions can be obtained by just allowing cycles of length 3 instead of allowing 2-cycles only [12]; in practice, a cycle length cap of 3 is typically used. Using a mixed integer program (MIP) where there is a decision variable for each cycle no longer than L and constraints that state that accepted cycles are vertex disjoint, combined with specialized branch-and-price MIP solving software, the (3-cycle) problem is solvable to optimality in practice at the projected steady-state nationwide scale of 10,000 patients [1]. In all our experiments, we use that algorithm as a subroutine.

A recent innovation in kidney exchange is *chains* [13, 9, 10]. Each chain starts with an *altruistic* donor—that is, a donor who enters the pool, without a candidate, offering to donate a kidney to any needy candidate in the pool. Chains start with an altruist donating a kidney to a candidate, whose paired donor donates a kidney to another candidate, and so on. Chains can be longer than cycles in practice because it is not necessary (although desirable) to carry out all the transplants in a chain simultaneously.[2] Already, chains of length ten or more have been reported in practice [10]. To our knowledge, all kidney exchanges in the US now use chains (in fact, the National Kidney Registry is using chains only and no cycles). In our experience, roughly 5% of the pool is altruistic.

There are many more feasible chains in a network than cycles—because one does not have to find a way to close a chain into a cycle. The straightforward way to incorporate chains into the optimizer is to add from the end of each potential chain a fake edge of weight 0 to every vertex that represents an altruist. This way, chains look exactly like cycles to the solver and are handled correctly. Unfortunately, due to the removal of the cap of 3 on cycle length, this approach does not scale even remotely to the nationwide level. Rather, it currently scale only to around 200 patients, depending on the cap on chain length. (Of course, if the chain length cap is lower than the cycle length cap, then chains do not significantly increase the complexity.)

3. NATIONWIDE KIDNEY EXCHANGE

Starting around 2003, several regional kidney exchanges have gone live in the US. Two examples include those run by the Alliance for Paired Donation and the Paired Donation Network. However, in 2008, the United Network for Organ Sharing (UNOS)—which controls all organ transplantation in the US—initiated the formation of a *nationwide* kidney exchange. The benefits of such a large-scale exchange are numerous (see, for instance, [5]), and it is ubiquitously accepted that one centralized exchange is better than fragmenting the market into separate exchanges. The UNOS nationwide kidney exchange pilot went live with 77 transplant centers in October 2010, and uses our algorithms and software to conduct a match run every month. Starting in May 2011, chains were incorporated into the UNOS pilot program. Currently, the cycle cap is 3, while the chain cap was 20 and is now being increased to infinity if it turns out to be computationally feasible.

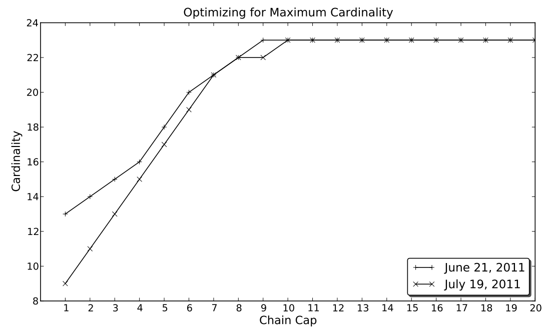


Figure 1: Real data from the June/July 2011 UNOS match runs, optimized for maximum cardinality.

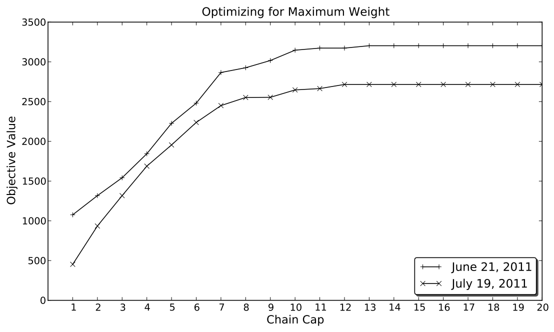


Figure 2: Real data from the June/July 2011 UNOS match runs, optimized for maximum total weight.

Figures 1 and 2 show results for two real matches, for June and July 2011. To show the efficacy of chains, we varied the chain cap from 1 (i.e., the altruistic donor donates directly to the deceased waiting list) to 20. In Figure 1, we maximize the cardinality of the final matching. That is, we ignore edge weights and assume all compatible matches are equally good, and determine the matching that allocates kidneys to the most candidates. The size of the matching increases significantly with chains up to length 9 (June) or 10 (July). Critically, with long chains we match 1.77 (June) and 2.55 (July) times the number of candidates than would have been matched with 3-cycles alone. We note that Ashlagi et al. [3] independently report similar findings from real-world data sets.

The improvement from long chains is even more drastic when the edge weights are taken into account, as is the case in the real UNOS match run. Figure 2 shows that in June, chains of length up to 13 increase the objective value, while chains of length up to 12 increase the objective of the matching in July. Overall, incorporating chains increases the objective value to 2.98 (June) and 6.00 (July) times that of chains only (with a cycle cap of 3).

It is important to note that the structure of the compatibility

[1]Such backing out cannot be prevented by legal means because it is illegal to contract for an organ in most countries.

[2]Unlike in a cycle, if a chain breaks by some donor backing out, the chain merely stops, but no patient-donor pair is out their "bargaining chip" (donor kidney).

graph, $G(n)$ in this early pilot program is special and, in many ways, computationally fortuitous. The current UNOS pool consists mainly of *highly sensitized* patients—that is, patients that are difficult to match based on their tissue type. Intuitively, these patients were too hard to match regionally and in prior runs of the national exchange—so the input graph is very sparse. Our other experiments have shown that with a less sensitized pool, we often cannot even solve the current problem size (with long chains) because the input graph $G(n)$ is not as sparse. Luckily, in the next section, we show theoretical results stating that in large kidney pools drawn from the full set of candidates (i.e., not just highly sensitized ones), long chains will have negligible effect on the overall cardinality of the matching with high probability. Therefore, one may not need to consider long chains in the clearing. This would be desirable in practice because short chains are (1) computationally dramatically more tractable for the clearing algorithm (there are fewer of them), (2) logistically easier to administer, and (3) less likely to fail due to a positive crossmatch or some non-simultaneous donor backing out (these two issues will be discussed later).

4. THEORETICAL BOUNDS ON CHAINS

In this section, we prove that using chains of length more than 3 provides no benefit in large, random, unweighted candidate pools. We will prove this result in the most common model of kidney exchange. We begin by describing the model.

4.1 Necessary background & model

The need for kidney exchange exists due to the myriad of immunological incompatibilities that can be present between a candidate and any potential donor. For instance, the *blood type* of a donor kidney can result in acceptance or outright rejection in a possible candidate. At a high level, human blood is split into four types—O, A, B, and AB—based on the presence or absence of the A and B proteins. While other complications may arise, a type O kidney can be transplanted into any candidate; type A and B kidneys can be transplanted into A and B candidates respectively, or an AB candidate; and type AB kidneys are limited to only type AB candidates. Therefore, some candidates are more difficult to match with a random donor than others. O-candidates are the hardest to match because only O-type kidneys can be given to them. Similarly, O-donors are the easiest to match.

With this in mind, candidate-donor pairs in the matching pool can be labeled based on their blood types using the *ABO model*; it is the *de facto* model for theoretical market design work on kidney exchange (see, e.g., [2, 5, 7, 15, 17]). An *under-demanded* pair is any pair such that the donor is not ABO-compatible with the candidate. Furthermore, if these pairs contain only type A and B blood (e.g., the candidate is type A and the donor is type B), the pair is called *reciprocal*. Any pair in the pool such that the donor is ABO-compatible with the candidate is called *over-demanded*. Furthermore, if a donor and candidate share the same blood type, they are a *self-demanded* pair. Intuitively, under-demanded and reciprocal pairs are "harder" to match than over-demanded and self-demanded pairs. In the ABO model, all compatible transplants are considered to be equally good (i.e., those edges have weight 1 each) and typically results in the ABO model are derived in the limit, when the number of pairs of each kind approaches infinity.

If blood type compatibility were the only requirement for a successful kidney donation, over-demanded and self-demanded pairs would have no need to enter the exchange pool because they could simply conduct the transplant within the pair. However, further complications force their hand: the people in a pair are usually incompatible due to tissue type. Tissue type, in particular what is known as HLA type, is measured as a combination of six proteins. Each potential candidate and potential donor must be tested for preformed antibodies against these six proteins; this needs to be done at least once a month because the antibody state of a person changes over time. An increase in the mismatches between donor and candidate HLA types decreases the likelihood of a successful kidney transplant, and can render a donor and candidate incompatible. These kinds of blood tests where measurements are taken separately from the donors and the patients are called *virtual crossmatch* for reasons that will become obvious in the next paragraph.

An important challenge is that medical knowledge is incomplete: even if a patient and donor are compatible based on the virtual crossmatch (so there is an edge in the input graph), in reality they might not be compatible (i.e., the edge might not be usable). This is determined days before the operation by conducting a test called a *crossmatch*: blood from the patient and blood from his/her planned donor are mixed together and if the mixture coagulates, they are incompatible. Such an unfortunate, but very common, occurrence is called a *positive crossmatch*. Positive crossmatch-sensitive models have only recently begun to appear in the literature, and have not included a study of chains [5, 15].

We will say that if an altruist donates directly to the deceased-donor waiting list, that constitutes a chain of length 1. If an altruist donates to a pair, whose donor donates to the deceased-donor waiting list, that constitutes a chain of length 2. If an altruist donates to a pair, whose donor donates to a pair, whose donor donates to the waiting list, that constitutes a chain of length 3, and so on. We are now ready to prove the main theoretical result of this paper.

4.2 Short chains suffice (in theory)

In this section, we use the canonical model for generating kidney exchange data [5]. It works as follows. We start with $G(n)$, a large compatibility graph representing a kidney exchange as described above. The set of n incompatible patient-donor pairs is partitioned into subsets $V_{X\text{-}Y}$ of type X-Y, for each combination of blood types X and Y of the patient and donor respectively. For each blood type X we denote the set of altruistic donors with that blood type by V_X, but make no assumptions about the size of these sets. We assume that a donor and a patient who are blood type compatible are tissue type incompatible with constant probability $\bar{\gamma}$, corresponding to the virtual crossmatch described above. The frequency of each blood type X is denoted by μ_X.

We are now ready to state our main theoretical result. It extends the recent results of Ashlagi and Roth [5] to the setting with chains.

THEOREM 1. *Assume that $\bar{\gamma} < 2/5$, $\mu_O < 3\mu_A/2$, and $\mu_O > \mu_A > \mu_B > \mu_{AB}$. Then with high probability $G(n)$ has an efficient allocation (i.e., one that saves as many patients as possible) that uses only cycles of length at most 3 and chains of length at most 3.*

The proof follows from three lemmas. The first lemma is a trivial simplification and extension of Lemma 9.5 of Ashlagi and Roth [5], which is a generalization of a classic theorem by Erdös and Rényi. To understand the lemma, denote by $G(n, p)$ a random graph with n vertices where an edge exists between two vertices with probability at least p. For a vector $\vec{\alpha} = (\alpha_1, \ldots, \alpha_r)$ where $\alpha_i \geq 0$ for $i = 1, \ldots, r$ let $G(\vec{\alpha}, n, p)$ be an r-partite graph with r sets of vertices $V_1, \ldots, V_r$ where $|V_i| = \alpha_i \cdot n$ for $i = 1, \ldots, r$, and a directed edge between $v \in V_i$ and $v' \in V_{i+1}$ for $i = 1, \ldots, r-1$, or between $v \in V_r$ and $v' \in V_1$, exists with probability at least p. A *perfect allocation* in a graph $G(n, p)$ matches all the vertices; a perfect allocation in $G(\vec{\alpha}, n, p)$ (consisting of cycles of length r) matches all the vertices in the smallest vertex set V_i for $i = \text{argmin}_j |V_j|$.

Deviating from [5], define $G'(\vec{\alpha}, n, p)$ similarly to $G(\vec{\alpha}, n, p)$, except that there are no edges between V_r and V_1. An allocation in $G'(\vec{\alpha}, n, p)$ consists of chains of length r that originate in a vertex in V_1. As before, a perfect allocation in $G'(\vec{\alpha}, n, p)$ matches all the vertices in the smallest vertex set V_i for $i = \text{argmin}_j |V_j|$.

LEMMA 1 (ASHLAGI & ROTH [5]). *Let $p > 0$. Then $G(n, p)$ admits a perfect allocation that uses cycles of length at most 3 with high probability. In addition, for any vector $\vec{\alpha}$ as above, the random graphs $G(\vec{\alpha}, n, p)$ and $G'(\vec{\alpha}, n, p)$ admit a perfect allocation with high probability.*

Using Lemma 1, we can assume that if we single out several large groups of vertices (in a large random compatibility graph) that correspond to blood type compatible pairs, there will be sufficiently many edges to admit a perfect matching. For example, if there are large sets of AB-O pairs, O-A pairs, and A-AB pairs, then with high probability we can find an allocation that consists of 3-cycles that matches all the vertices in the smallest set. Even if we consider several such allocations sequentially, by applying the union bound we can see that they all exist with high probability. This essentially allows us to assume in the proof of the next lemma that any two vertices that are blood type-compatible are connected by an edge.

LEMMA 2. *Let $G(n)$ be a random graph that admits the following allocation:*

1. *Every self-demanded pair is matched in 2-way or 3-way cycles with other self-demanded pairs.*
2. *Every B-A pair is matched in a 2-way cycle with an A-B pair.*
3. *Every A-B pair that is not matched to a B-A pair is matched in a 3-way cycle with an O-A pair and an A-AB pair.*
4. *For $X \in \{A, B\}$, every over-demanded pair X-O is matched in a 2-way cycle with an O-X pair.*

Then with high probability $G(n)$ admits an efficient allocation that uses cycles of length at most 3 and chains of length at most 3.

PROOF SKETCH. We complete the allocation described in the lemma's statement to an efficient allocation. Figure 3 visualizes the augmented allocation; regular edges are assumed by the lemma's formulation while dashed edges are added during this proof. Let V^1 be the set of vertices not matched by the initial allocation. First, as many A-donors as possible donate to A-AB pairs and as many B-donors as possible donate to B-AB pairs (shown in Figure 3 by dashed edges from A-altruists to A-AB pairs and from B-altruists to B-AB pairs). In both cases, one of the two vertex sets will be exhausted. More formally, using Lemma 1 we find a perfect allocation for the subgraph induced by V^1_A and $V^1_{\text{A-AB}}$, and similarly we find a perfect allocation for the subgraph induced by V^1_B and $V^1_{\text{B-AB}}$.

Let V^2 be the vertices not matched by previous allocations. We find as many 3-way $(\text{AB-O}, \text{O-A}, \text{A-AB})$ cycles as possible, that is, we find a perfect allocation for the subgraph induced by $V^2_{\text{AB-O}}$, $V^2_{\text{O-A}}$, and $V^2_{\text{A-AB}}$. It may be the case that $V^2_{\text{A-AB}} = \emptyset$. Let V^3 be the set of vertices not matched by previous allocations. Next we find a perfect allocation with 3-way $(\text{AB-O}, \text{O-B}, \text{B-AB})$ cycles. It may be the case that $V^3_{\text{AB-O}} = \emptyset$ or $V^3_{\text{B-AB}} = \emptyset$.

Let V^4 be the vertices not matched by previous allocations. The next component in the constructed allocation matches as many O-donors as possible in chains of length 3 of the form $(\text{O}, \text{O-A}, \text{A-AB})$ and then $(\text{O}, \text{O-B}, \text{B-AB})$. This is done sequentially as above. Finally, we match the remaining O-donors and AB-O pairs with remaining under-demanded pairs via chains of length 2 or 2-way cycles (not shown in Figure 3).

Each of the allocations constructed above exists with high probability; thus (by applying the union bound) they all exist with high

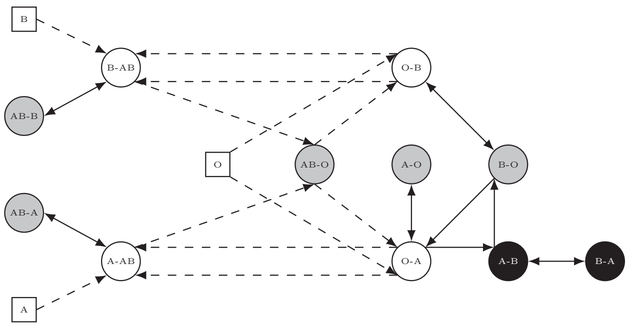


Figure 3: Accompanying figure to Lemma 2. Altruists are shown as rectangles; candidate-donor pairs as ovals. Over-demanded pairs are gray, under-demanded are white, and reciprocal pairs are black. Regular edges appear in the lemma's formulation and dashed edges are constructed in the proof.

probability. To complete the proof, we argue that our construction gives rise to an efficient allocation. Since under our construction all over-demanded, self-demanded, and reciprocally demanded pairs are matched, it is sufficient to show that no allocation can match more under-demanded pairs.

Following Ashlagi and Roth [5], when vertex v participates in an exchange with under-demanded vertex v' we say that v *helps* v'. Self-demanded and reciprocally demanded pairs cannot help under-demanded pairs without involving donors or over-demanded pairs. Similarly, AB-donors cannot help under-demanded pairs. In addition, only two types of vertices can help two under-demanded pairs: AB-O pairs can participate in cycles with one of O-A and O-B and one of A-AB and B-AB, and O-donors can start a chain with the same types. Any other vertex can help at most one under-demanded pair, and in particular over-demanded pairs of type X-$Y \neq$ AB-O can only help under-demanded vertices of type Y-X.

Now, A-donors can only help A-AB pairs, and B-donors can only help B-AB pairs. Therefore, it is optimal to match these donors with their respective under-demanded pairs. Finally, in our constructed allocation as many AB-O pairs and O-donors as possible are helping two under-demanded pairs each, while the rest are helping one under-demanded pair each. $\square$

The following lemma directly follows from Proposition 5.2 of [5], and holds under the assumptions of Theorem 1.

LEMMA 3 (ASHLAGI & ROTH [5]). *$G(n)$ has an allocation as in Lemma 2, up to symmetries between A-B pairs and B-A pairs, with high probability.*

4.3 Discussion

Theorem 1 follows from the proofs of the three lemmas in Section 4.2. The theorem itself is motivated by the recent work of Ashlagi and Roth [5]. One has to be careful, though, not to use the exact allocation constructed in Proposition 5.2 of their paper as a starting point for the efficient allocation that involves altruistic donors. Indeed, given that $|V_{\text{A-B}}| \geq |V_{\text{B-A}}|$, Ashlagi and Roth match AB-O pairs in cycles $(\text{AB-O}, \text{O-A}, \text{A-AB})$. However, because we are essentially making no assumptions regarding $|V_A|$ and $|V_B|$, it may be the (admittedly extreme) case that there are many (say an infinite supply) of A-donors, few B-donors, few O-donors, and a large number of unmatched under-demanded pairs of type O-B and B-AB. In that case we would rather have the A-donors donate to A-AB pairs while creating cycles $(\text{AB-O}, \text{O-B}, \text{B-AB})$. Therefore, we must match AB-O pairs only *after* matching altruistic donors.

The presence of (even short) chains allows us to avoid a negative property of the efficient allocation constructed by Ashlagi and Roth [5]: that it never matches O-AB pairs. These are, in a sense, the "most" under-demanded pairs in that their candidates are hardest to match, while their donors are least capable of finding a match. In our allocation, AB-O pairs and O-donors that cannot participate in 3-cycles can donate to O-AB pairs without affecting the size of the matching. More precisely, if there are sufficiently many donors to fully match one of the sets $V_{\text{O-A}}$ and $V_{\text{A-AB}}$, and one of the sets $V_{\text{O-B}}$ and $V_{\text{B-AB}}$, then an efficient allocation can match O-AB pairs.

Independent work by Ashlagi et al. [3] attempts to explain the observed benefit of longer chains by considering a theoretical model with highly sensitized patients. Specifically, the probability of tissue type compatibility is allowed to decrease with the size of the graph n. Among other results, it is shown that for any k there exists a small enough probability of compatibility such that chains of length $k+1$ are strictly better than chains of length k. However, to even derive such a statement for chains of length 5 versus chains of length 3, the probability must be as small as c/n for some constant c, whereas intuitively this probability should be a constant that does not depend on n. Hence, despite the elegance of their results, the assumptions underlying their model may be hard to justify.

5. EXPERIMENTAL VALIDATION

The theoretical results from Theorem 1 are strong in that they limit the utility of chains to those of length 3 or fewer—as the graph grows to infinity. In this section we study the disconnect between that theorem and the real-world results from the recent UNOS kidney match runs (Figures 1 and 2).

There are three potential reasons for this disconnect: (1) the theory applies in the large, and the UNOS exchange is not yet large enough for the theory to have taken hold, (2) the model that each blood type compatible edge fails tissue type compatibility independently and with equal probability is a poor model of the (highly sensitized) UNOS pool, and (3) the theory assumes all edges have equal weight, while in the UNOS exchange, edges are weighted.

The discrepancy between the theory and the fielded results cannot be explained solely by the fact that the theory model uses unweighted edges while the real UNOS data has edge weights. If that were the main difference, we would see the curves in Figure 1 reach their maxima at a chain cap of 3. This is not the case. So, we see that even if all the weights were binary, long chains would produce a significant benefit in practice. The difference can, in part, be attributed to the highly structured and very small UNOS pool. This is the product of the newness of the UNOS pilot program; as the exchange grows, we expect the compatibility graph's structure to converge to one similar to our theoretical model.

In reality, the input graph $G(n)$ cannot grow infinitely; specifically, in kidney paired donation, it has been estimated that in steady state the fully fielded nationwide exchange will have around 10,000 pairs at any one time. In this section, we experimentally determine just how large the candidate pool needs to be for the chain cap prescribed by Theorem 1 to apply.

The minimum size of this compatibility graph needed for the theory to take hold depends on the probability distribution of blood and HLA types in the candidate and altruist pools, the number of candidates in the graph, and the number of altruists. We will vary both the number of candidates and altruists, but choose to focus only on blood and HLA types representative of the US population (which serves the current nationwide kidney exchange).

Here we generate candidate-donor pairs and altruists via the most advanced and commonly used data generator for kidney exchange today, by Saidman et al. [14]. This generator incorporates the blood types from the ABO model discussed earlier. It also incorporates an abstract model of tissue types to compute a type of score that quantifies the likelihood of a specific candidate being tissue type compatible with a random donor. In other words, this tissue type model is more refined than assuming all blood type compatible edges are tissue type incompatible with equal probability.

5.1 Increasing the candidate pool size

In the first set of experiments, we explore the effect of a large number of *candidates* on the efficacy of long chains. We hold the number of *altruists* constant at 1, 5, or 10 for each experiment.

Figures 4, 5, and 6 show that larger pools match a higher percentage of candidates, leveling out at roughly 62% in compatibility graphs with a couple hundred candidates. At a high level, this is a strong argument for a national kidney exchange to replace the set of smaller regional exchanges; see [11] for similar arguments. These figures also make a case for the inclusion of chains in pools at both the regional and national level. Figure 5 shows that, for generated pools of size 256, the optimal matching with a chain cap of 1 (i.e., altruists donating directly to the deceased waiting list, avoiding the paired candidate pool entirely) matches nearly 4% fewer candidates overall than matching with a chain cap of 3. The case is more drastic as the number of altruists increases; for instance, Figure 6 shows a 5% decrease on compatibility graphs of the same size. The effect of altruists on the pool is discussed further in the next section.

From above, we can now ignore matchings that only include chains of length 1 and 2; capping chains at either of these levels would result in fewer candidates being matched. Figures 7, 8, and 9 show the expected number of extra transplants resulting from matches incorporating chains of length 4 and 5, compared to only considering chains of up to length 3. Clearly, the maximum number of additional transplants offered by increasing the chain cap by 1 is proportional to the number of altruists present in the graph. For example, for a graph with a altruists, incorporating 5-chains can provide a benefit of at most $2a$ matches over incorporating at most 3-chains; similarly, increasing the cap from 3 to 4 results in at most a extra matches. Figures 7 and 8 show that at pool sizes of 256 with $a = 1$ and $a = 5$, the expected number of additional transplants for either 4- or 5-chains is nil (over 100 generated compatibility graphs). Figure 9 shows similar results while exemplifying another behavior: as the number of altruists increases, the size of the pool required so that limiting the mechanism to 3-chains is satisfactory increases. This behavior is explored further in the next section.

Figures 8 and 9 initially show an *increase* in the utility of longer chains as the graph size moves from very small (e.g., 16 candidates) to slightly larger (e.g., 32–64 candidates).[3] This is a side effect of the number of altruists present relative to the size of the pool. With a high enough ratio of altruists to candidates, altruists can "flood" the matching, an idea explored further in the next section.

All of the experiments validate the theory: there seems to clearly be a pool size beyond which long chains do not help.

5.2 Increasing the number of altruists

In the previous subsection, we held the number of altruists constant while increasing the size of the candidate pool. We now explore the opposite, allowing ever increasing numbers of altruists to enter candidate pools of constant size.

As the number of altruists increases relative to the size of the candidate pool, the expected number of candidates matched rises to 100%, as shown in Figures 10, 11, and 12. This full flooding of the pool to create a complete matching, while interesting, is not

[3]In Figure 9, the computational demands of this experiment precluded us from extending the dotted line past 128 candidates.

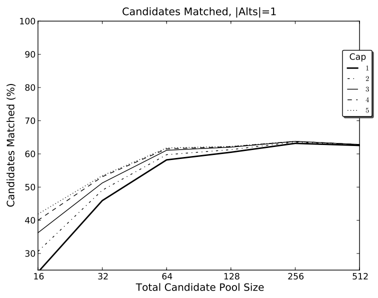


Figure 4: Total percentage of candidates matched as #candidates increases across various chain caps, #altruists=1.

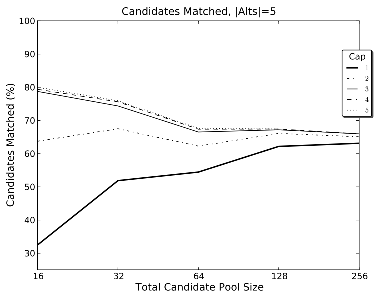


Figure 5: Total percentage of candidates matched as #candidates increases across various chain caps, #altruists=5.

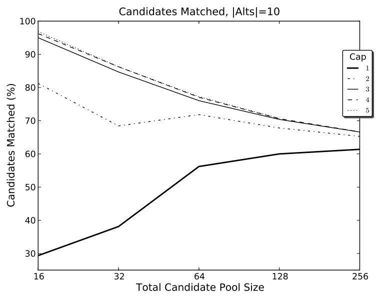


Figure 6: Total percentage of candidates matched as #candidates increases across various chain caps, #altruists=10.

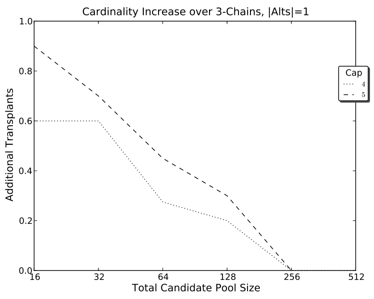


Figure 7: Cardinality increase over 3-chains for 4- and 5-chains, #altruists=1.

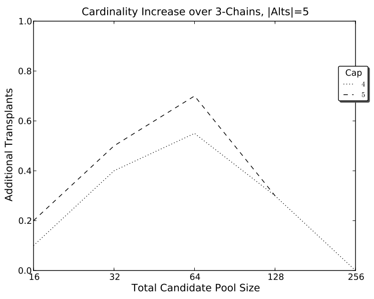


Figure 8: Cardinality increase over 3-chains for 4- and 5-chains, #altruists=5.

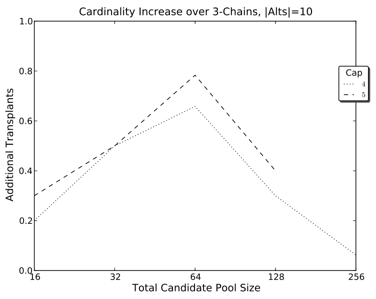


Figure 9: Cardinality increase over 3-chains for 4- and 5-chains, #altruists=10.

presently a realistic scenario; all three tested compatibility graph sizes would require around 50% as many altruists as candidates in the pool (Figure 12 has the x-axis cut short). In our experience with UNOS, the number of altruists is typically around 5% the size of the candidate pool. Increasing this number could feasibly change as the exchange grows in size and publicity, paying special notice to the ethical issues that arise in coercion of possible donors.

6. DYNAMIC KIDNEY EXCHANGE

In the paper so far, we have studied static models. We now discuss the dynamics of a kidney exchange running month to month.

6.1 Augmenting the model

We augment the model in several ways to make it capture the nuances that have arisen in practice.

Dynamics. Most of the work in kidney exchange has focused on a single-shot optimization on a static pool. This deviates from reality in that matching should occur *dynamically*. In reality, candidates arrive and depart from the pool. Even with dialysis, only 12% of patients survive 10 years [18]; this gives us the monthly death rate we use in our experiments. Timeliness in matching is clearly important. Our experimental results, discussed later, perform matching over 24 months using a changing kidney pool.

Some work in this area has been done already. Ünver [17] derives an efficient mechanism in the dynamic setting for a simplified model of kidney exchange that can be solved analytically. Awasthi and Sandholm [6] apply the model discussed above to the dynamic setting, using trajectory-based optimization to look into the possible futures and then use optimization technology to determine transplants for the current period, including chains.

Work by Gentry et al. [8] on simulated data and Ashlagi et al. [4] on real-world data explores the trade-offs between two types of chain execution polices. The first chain type is executed in its entirety in one time period, with the leftover donor donating to the waiting list. An alternative is to split long chains into segments with intra-segment simultaneous transplants, but the segments execute one after another. The left over donor (aka *bridge donor*) from one segment then serves as a virtual altruist for the next segment. These two types of chains perform differently under the presence of *renege rates*—that is, when a bridge donor decides to leave the pool before donating a kidney. However, no reliable quantification of a renege rate exists due to the infancy of kidney exchanges.

While Gentry et al. [8] do not explicitly consider chain caps, Ashlagi et al. [4] do; they experimentally show that longer (up to length 6) chains can, in fact, help. Our work uses a similar model with single-shot execution chains and, importantly, takes into account the policies of the UNOS nationwide kidney exchange. As we will show, this addition results in different matching behavior. We now discuss these UNOS-specific additions to the model.

Individual crossmatch sensitivity. As exemplified in the real, highly-sensitized UNOS candidate pool, candidates can have widely varying susceptibility to incompatibilities in kidney donation. The Saidman et al. model from the previous section has a rather realistic view of virtual crossmatch failures, and we use that model here.

In addition, here we do (non-virtual) crossmatches for all the planned transplants just before the transplant takes place, as in reality. This is again done using the Saidman et al. [14] generator. It provides for each candidate a probability that the candidate is tissue type compatible with a random person. We use that probability to draw crossmatch success versus failure. If the crossmatch fails, the transplant cannot proceed. If it is part of a cycle, the cycle does not execute; the pairs in the cycle go back in the pool. The failed edge is permanently removed from the compatibility graph $G(n)$.

Crossmatching has a significant effect on the size of the "real" matching. Assume an optimal matching (pre-crossmatch) yields a 3-cycle. If *any* crossmatch fails between a candidate and potential donor, the *entire* cycle must be thrown away—since we cannot force a donor to give a kidney if his accompany candidate does not

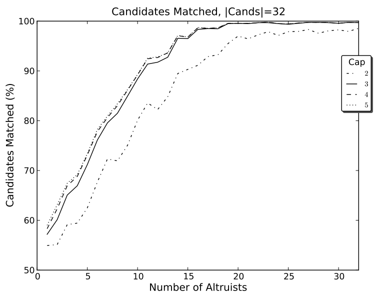


Figure 10: Total percentage of candidates matched as #altruists increases across various chain caps, #cands=32.

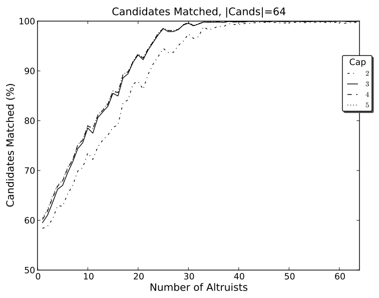


Figure 11: Total percentage of candidates matched as #altruists increases across various chain caps, #cands=64.

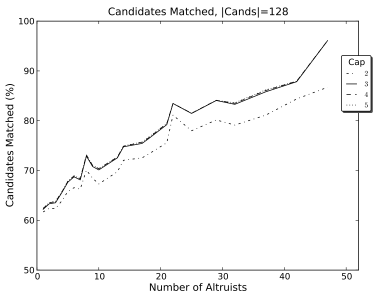


Figure 12: Total percentage of candidates matched as #altruists increases across various chain caps, #cands=128.

receive one. Even more drastic is the case of chains: if, for example, a pre-crossmatch matching yields a 20-chain, any transplants after the first crossmatch failure cannot be performed.

Because of this special case for chains, real-world exchanges have enacted policies for the acceptance or rejection of chains based on their length and the quality of the altruistic donor. O-type altruists are highly valued, as they can (potentially) donate to any blood type, so short chains enabled by O-type altruists should (potentially) be rejected in favor of longer chains in the future. Our experiments follow current UNOS policy which, along with some special cases discussed below, states that (i) chains started by non-O-type altruists are always executed, while (ii) chains triggered by an O-type altruist are executed only if they can be executed to length at least 5 (before there is a crossmatch failure). We will experiment with varying the value away from 5; we will call this parameter k.

Altruists are allowed choices. In the event that an O-type chain is shorter than length 5, the UNOS policy allows for the altruist to decide that the chain be executed anyway. This is due to the fact that altruists do not want to stay in the candidate pool indefinitely, but rather want to move on with their lives and other plans. In UNOS's experience running kidney exchange, altruists typically do not wish to stay active in the pool for more than three months—instead opting to donate directly to the deceased donor waiting list. While exact data on this phenomenon are too sparse at the moment, our experiments use the anecdotal rates (received through UNOS): 75% probability of an altruist requesting execution of a short chain, and a monthly altruist exit rate that corresponds to an expected presence of two months in the pool for each altruist. Our model executes each chain in a single time segment.

6.2 Experimental Results

We now present preliminary results simulating dynamic kidney exchange under the model described above. Figure 13 shows the expected increase in transplants when including chains over the cycles-only approach. The x-axis describes the total number of candidates available during at least one time period over the entire simulation; between 15 and 20 candidates arrive every time period and between 1 and 2 altruists arrive every time period. The initial pool (i.e., the pool at time $t = 0$) is seeded with between 50 and 100 candidates and 5 altruists. These settings roughly mimic the current state of the nationwide UNOS pilot program.

The results both remain true and (appear to) deviate from the theory in a number of ways. The benefit of using chains is immediately obvious; in all cases, even using only 2-chains increases the total number of transplants by 20 or more. However, in this new setting, chains of length at most 3 (at least for the tested pool sizes, number of altruists, etc) do *not* provide equivalent benefit to longer chains. While 3-chains do provide a net gain over 2-chains, con-

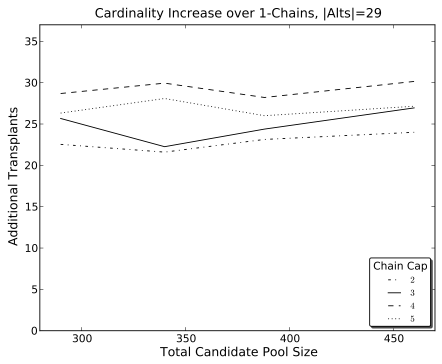


Figure 13: Expected improvement of n-chains over 1-chains (over 24 months).

sidering longer chains helps—sometimes by nearly 10 additional transplants. This increase is surprising because, intuitively, longer chains are less likely to be executed in full (and thus likely to be canceled by the UNOS policy) due to low crossmatch probability. Not executing a chain is dangerous because altruists leave the pool entirely if they remain unmatched for more than a few months.

The results above can be explained by considering the effect of time on an evolving small-scale pool of candidates. Over time, highly sensitized candidates will build up in the pool, since they are often significantly harder to match—both because they have fewer connected edges in the generated compatibility graph and because they are more likely to fail during the crossmatch. Through the real-world results detailed in Section 3, we have seen that the utility of (long) chains increases tremendously in the presence of a small, highly sensitized pool. In Figure 13, chains of length greater than 3 are able to serve highly sensitized candidates because they do not need to "close" the chain, as is the case with a cycle.

Surprisingly, allowing the optimizer to use chains of up to length 5 is strictly worse than constraining it to chains of length at most 4 (while a cap of 4 is better than 3). This suggests that there is diminishing benefit to longer and longer chains, and at the same time there is increasing risk of crossmatch failure (and thereby altruists leaving and candidates dying) with increasing chain cap. The experiments here suggest that in the dynamic setting with these pool sizes (i.e., not in the very large), a chain cap of 4 is best.

We now expand our preliminary experiments to include the chain execution policy from UNOS (see Section 6.1), and we will vary k (between 1 and the chain cap). Intuitively, a higher k will prevent "wasting" a valuable O-altruist on short chains, favoring waiting for a longer, higher-scoring chain instead. Figure 14 shows the effect of varying k as we increase the chain length cap. When considering only short chains, a higher k increases the total number of transplants. In contrast, when chains of length 4 and 5 are considered, it appears better to reduce k. The drop in overall utility from allowing *only* long chains to execute is due to altruists' propensity to leave the pool; if an altruist is not used in an executed chain within a few time period, he/she is likely to leave the pool (and thus be "wasted" by going straight to the deceased donor waiting list instead of saving some lives in the pool first).[4]

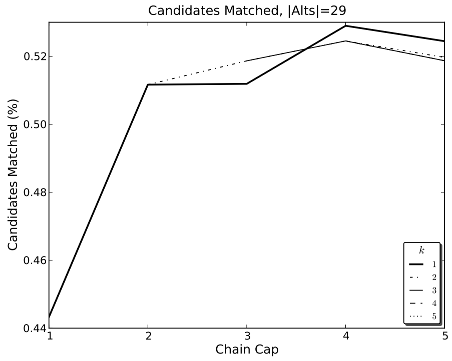


Figure 14: Expected percentage of candidates matched with 1- to 5-chains, varying k in the UNOS chain execution policy.

7. CONCLUSIONS & FUTURE RESEARCH

In this paper, we considered altruist-initiated chains, a recent innovation in barter exchanges that has seen wide adoption in regional and national kidney exchange, but has not been well understood. We described results gathered from the first nationwide kidney exchange in the US that show, for relatively small, highly-sensitized pools of candidates, the benefit of long chains. We then showed that, in the large, the benefit from chains longer than 3 becomes negligible (with high probability) on random compatibility graphs drawn from distributions that mimic the real world population. We supported these theoretical results by extensive experiments using the state-of-the-art instance generator to allow us to experiment on larger instances than exist in current kidney exchanges. The theoretical results take hold in exchanges orders of magnitude smaller than the expected steady-state of the nationwide kidney exchange; this provides evidence for considering only short chains in the large, real-world exchanges we expect to see.

Finally, we experimented in the dynamic setting where the exchange clears every month. We included in the simulations all the known (to us) considerations that have arisen through our work with real kidney exchanges. Computational complexity precluded experiments in the large for the dynamic setting, but in medium-sized pools a chain cap of 4 was best (and strictly better than 5). At any given point in our largest dynamic simulations, 100–150 candidates were present in the pool—others had already been matched, had died, or had not entered the simulation yet. We showed in Section 5 that, at such a small size and with so many altruists, we cannot expect 3-chains to suffice. We believe that, were the pool increased to hundreds of new candidates per month (as is projected to be the case in a fully fielded nationwide exchange), experiments in a dynamic setting would yield results similar to the static setting—with chains of length 3 sufficing.

Many avenues for future research arise from this work. Theoretical results in less abstract models would provide further insight into the efficacy of chains in real world exchanges. Ongoing work by Ashlagi et al. [3] is, to our knowledge, the only other push in this direction; they analyze chains in highly sensitized pools, but under arguable assumptions. Furthermore, advances in clearing algorithms are necessary to handle chains at even the moderate scale; the current state of the art can clear only small candidate pools with just a few altruists. Scaling to the expected size of the nationwide kidney exchange will require algorithmic and computational advances that allow clearing pools orders of magnitude larger than what can be solved today. Restricting attention to short chains may be a promising avenue for tackling that complexity.

Acknowledgments. This work is supported by the National Science Foundation under grants IIS-0905390, IIS-0964579, and CCF-1101668. We acknowledge Intel Corporation and IBM for gifts.

[4]Following the acceptance of this paper, UNOS removed the rule that chains triggered by an O-type altruist are executed only if they can be executed to length at least 5. Our experimental results were the reason for this change in policy.

8. REFERENCES

[1] D. Abraham, A. Blum, and T. Sandholm. Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. *EC*, 2007.

[2] I. Ashlagi, F. Fischer, I. Kash, and A. Procaccia. Mix and match. *EC*, 2010.

[3] I. Ashlagi, D. Gamarnik, and A. Roth. The need for chains in kidney exchange. *NBER Market Design Workshop*, 2011.

[4] I. Ashlagi, D. Gilchrist, A. Roth, M. Rees. Nonsimultaneous Chains and Dominos in Kidney-Paired Donation—Revisited. *Am J Transplant*, 2011.

[5] I. Ashlagi and A. Roth. Individual rationality and participation in large scale, multi-hospital kidney exchange. *EC*, 2011.

[6] P. Awasthi and T. Sandholm. Online stochastic optimization in the large: Application to kidney exchange. *IJCAI*, 2009.

[7] I. Caragiannis, A. Filos-Ratsikas, and A. Procaccia. An improved 2-agent kidney exchange mechanism. *WINE*, 2011.

[8] S. Gentry, R. Montgomery, B. Swihart, D. Segev. The roles of dominos and nonsimultaneous chains in kidney paired donation, *Am J Transplant*, 2009.

[9] R. Montgomery *et al*. Domino paired kidney donation: a strategy to make best use of live non-directed donation. *The Lancet*, 2006.

[10] M. Rees, J. Kopke, R. Pelletier, D. Segev, M. Rutter, A. Fabrega, J. Rogers, O. Pankewycz, J. Hiller, A. Roth, T. Sandholm, U. Ünver, R. Montgomery. A nonsimultaneous, extended, altruistic-donor chain. *New Engl J Med*, 2009.

[11] A. Roth, T. Sönmez, and U. Ünver. Kidney exchange. *Q J Econ*, 2004.

[12] A. Roth, T. Sönmez, and U. Ünver. Efficient kidney exchange: Coincidence of wants in a market with compatibility-based preferences. *Am Econ Rev*, 2007.

[13] A. Roth, T. Sönmez, U. Ünver, F. Delmonico, L. Saidman. Utilizing list exchange and nondirected donation through 'chain' paired kidney donations. *Am J Transplant*, 2006.

[14] S. Saidman, A. Roth, T. Sömnez, U. Ünver, F. Delmonico. Increasing the opportunity of live kidney donation by matching for two and three way exchanges. *Transplantation*, 2006.

[15] P. Toulis and D. Parkes. A random graph model of kidney exchanges: efficiency, individual-rationality and incentives. *EC*, 2011.

[16] United Network for Organ Sharing. http://www.unos.org/.

[17] U. Ünver. Dynamic kidney exchange. *Rev Econ Stud*, 2010.

[18] United States Renal Data System. http://www.usrds.org/.

Fair Allocation Without Trade

Avital Gutman [*]
The Hebrew University of Jerusalem
gutmant@cs.huji.ac.il

Noam Nisan[*]
The Hebrew University of Jerusalem
noam@cs.huji.ac.il

ABSTRACT

We consider the age-old problem of allocating items among different agents in a way that is efficient and fair. Two papers, by Dolev et al. and Ghodsi et al., have recently studied this problem in the context of computer systems. Both papers had similar models for agent preferences, but advocated different notions of fairness. We formalize both fairness notions in economic terms, extending them to apply to a larger family of utilities. Noting that in settings with such utilities efficiency is easily achieved in multiple ways, we study notions of fairness as criteria for choosing between different efficient allocations. Our technical results are algorithms for finding fair allocations corresponding to two fairness notions: Regarding the notion suggested by Ghodsi et al., we present a polynomial-time algorithm that computes an allocation for a general class of fairness notions, in which their notion is included. For the other, suggested by Dolev et al., we show that a competitive market equilibrium achieves the desired notion of fairness, thereby obtaining a polynomial-time algorithm that computes such a fair allocation and solving the main open problem raised by Dolev et al.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*; J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*

General Terms

Algorithms, Economics, Performance

Keywords

Fisher Market, Fair Allocation, Leontief, Perfect Complementarity

1. INTRODUCTION

This paper deals with the classic question of allocating resources among different potential agents. Specifically, we are interested in this question in the context of computer systems that need to share their computational resources among different agents. Resources in this context can be CPU time, main memory, disk space, communication links, etc. The agents may be jobs, computers, or software agents representing them, and the allocation may be implemented at the level of the network routers, the operating system, or by higher level software, whether centralized or distributed.

The departure point of this work is several recent attempts to look at the allocation problem in computer systems in abstract principled terms; by Dolev et al.[6] and Ghodsi et al.[9]. In these papers, the basic model assumed that each agent desires a well-defined bundle of resources, and the allocation problem is to decide which fraction of his bundle each agent gets. While the treatment is abstract, it is very clear that the motivation came directly from actual computer systems. We wish to explicitly point out a key difference between the literature on allocating resources in computer systems and the general economic literature on resource allocation, a difference we believe explains the near complete separation between the two: The computing literature almost always assumes that each agent desires a well-defined bundle of resources, while the economic literature almost always considers the *trade off* that agents have between different resources.

The standard example of resource allocation by an operating system has each job requesting a well-specified set of resources (e.g. 1000 CPUs with 1TB of main memory and 1GB/sec of communication bandwidth), and allocates among such requests. We do not often see systems that can handle requests like "either 1000 CPUs with 2TB main memory or 2000 CPUs with 1TB main memory". In fact, even when the underlying problem allows a trade-off between several possible bundles of resources, the allocation system usually first decides on a bundle for each agent and then attempts to allocate these chosen bundles to all agents. An example is routing in a network, where the routing decision of choosing a path to the destination is in practice completely decoupled from the bandwidth-allocation decision for the links on the chosen path.

On the other hand, the economics literature on resource allocation usually focuses on the trade-offs between different resources that are captured by agent preferences. The fact that for some agent an apple may be a substitute to an orange results in a flexibility in preferences that allows sophisticated trade that can be beneficial to all parties (Figure 1a).

The case where consumers' preferences do not allow any substitution between different goods is called the case of "perfect complements" (Figure 1b), with the basic example being Leontief utilities: For any quantity vector $(x_1, ..., x_m)$ of m resources, u is defined as $u(x_1...x_m) = min_j(x_j/r_j)$, where the r_js are the relative proportions needed of the different goods [4]. These are the utilities used (implicitly) by [6, 9] to capture the preferences in computer systems, and are the focus of attention of this paper. However, the relations between resources are not necessary linear. It could be that some agents' demand of bandwidth would be in quadratic re-

[*]Supported by a grant from the Israeli Science Foundation (ISF), and by the Google Inter-university center for Electronic Markets and Auctions.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

lation to number of CPUs, and that RAM would be proportional to $\log(DISK)$. The generalization to perfectly complementary functions is required to capture such relations.

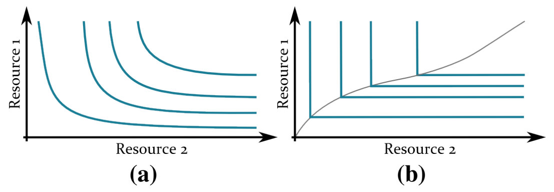


Figure 1: Figure 1a shows an indifference map for a utility function with some degree of substitution between goods. Figure 1b shows the indifference map of a perfectly complementary utility. Each indifference curve is the set of all allocations resulting in the same utility.

It is important to state that throughout this work, we do not assume or make any *interpersonal comparisons of utility* [10] - we use utility functions solely as a way to define agents' preferences over bundles.

Our first contribution in this paper is putting the question of allocation of computer resources studied in [6, 9] into an economic framework, obtaining a general economic perspective that we believe is useful. We observe that when agents have perfectly complementary preferences, the requirement of Pareto-efficiency turns out to be quite weak and simple: it suffices that the allocation is (what we term) *non-wasteful*: no agent receives resources that he has no use for, and no useful resources are left on the table. As a result, efficiency does not require any form of trade between agents, and there is no need for "money" as a mechanism of ensuring efficiency. This underlying lack of trade may explain the applicability to computer systems, which usually lack the infrastructure for enabling trade. It may also suggest potential applicability in other scenarios where trade is impossible due to technical, administrative, legal, or ethical reasons.

However, this requirement alone leaves us with multiple possible allocations (as seen in Figure 2b as opposed to Figure 2a). Therefore, the question of choosing between the efficient allocations becomes the central one, as indeed was done by [6, 9] in different ways, using different terminology and definitions of fairness.

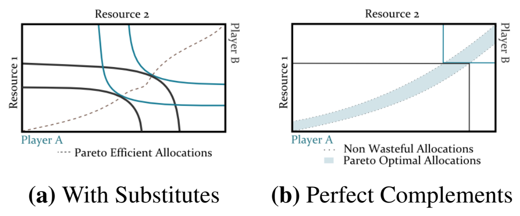


Figure 2: Figure 2a is an Edgeworth box of continuous indifference curves with substitutes. In this case, if two curves are tangent to each other, there is a single point of tangency. As a result, the Pareto set is single dimensional.
Figure 2b is an Edgeworth box with curves corresponding to the perfectly complementary case. Here, two curves may have many points of tangency, corresponding to allocations of the under-demanded goods. As a result, the Pareto efficient allocations are in the whole shaded area.

After formalizing our notions, which generalize those in [6, 9] to the framework of perfectly complementary utilities, we embark on the study of these fairness notions "Bottleneck Based Fairness" (BBF), advocated by [6], and "Dominant Resource Fairness" (DRF), used as the fairness property of the mechanism in [9].

Behind BBF there is the notion that when each agent is entitled to some percent of all resources, he has a "justified complaint" against an allocation if he gets less than that percentage on all "bottleneck" resources. An allocation is called BBF if no agent has a justified complaint against it. Regarding DRF, we view the notion as composed of two elements: The first element is a norm-like mechanism that quantifies and allows comparing bundles of resources allocated to a single agent. The second element dictates that fairness is defined according to Rawlsian [12] max-min fairness of these quantities. Meaning, fairness should be decided by the agent that received the least according to the quantification of bundles. DRF uses the L_∞ norm (i.e. looking at the highest allocated share over all resources); we note any norm is equally viable. This general norm-like mechanism defines a class of fairness notions that we call "Generalized Resource Fairness" (GRF) (see examples in Figure 3). A somewhat related generalization of DRF can be found in [11].

At this point come our main technical results - both quite simple given the wider context which we have built. We present two algorithms for finding fair allocations according to each of these fairness notions. Our first algorithm finds an allocation satisfying any given fairness notion from the GRF class. It is similar, in concept, to the family of "water-filling" algorithms [5, 2]. We extend the observation of [9] that greedily allocating the resources to the "poorest" agent at each stage produces a fair allocation. In [9] this was done in a pseudo-polynomial way (allocating δ-by-δ fractions of the goods[1]) for DRF and Leontief utilities. We give an algorithm which is polynomial for a wide subclass of perfectly complementary utilities and any GRF notion of fairness (given minimal access to an oracle describing the utilities and desired fairness-norm), and becomes strongly polynomial for Leontief utilities.

Our second algorithmic result solves the main problem left open by [6]: obtaining a polynomial-time algorithm for computing a BBF allocation. Our contribution here is a direct corollary of phrasing the problem in a market context, in which each agent enters the market with some quantity of each good (his endowment), or with a budget (some amount of money). We recast the entitlements of [6] in terms of budgets in a Fisher market[2] (see e.g. [4]), and look at competitive market equilibria of this market. Our main observation is then that any market equilibrium in this context will be BBF. This both proves the existence of a BBF allocation for all perfectly complementary utilities, and solves the main problem of [6] for Leontief utilities, for which an equilibrium in a Fisher market can be found in polynomial-time using convex programming [3]. Note that Ghodsi et al. [9] already compared their results with a market equilibrium.

The structure of the rest of the paper is as follows: Section 2 sets our notions, notations, and basic facts about efficiency with perfectly complementary preferences. In Section 3 we define and discuss various notions of fairness of allocation in this context, in Section 4 we present the algorithm for GRF fairness and in Section 5 we develop the market context which implies the algorithm for BBF.

2. THE MODEL

2.1 Some Notation

We use the following notation: For $\vec{x}, \vec{z} \in \Re^m$ -

- We write $\vec{x} \leqslant \vec{z}$ if $\forall j, x_j \leqslant z_j$.

[1][9] also suggest a polynomial-time algorithm which provides DRF for a discrete setup, a case which we do not study here.

[2]In terms of an Arrow-Debreu market [1], this is the scenario in which agents have equal endowments of all goods.

- We write $\vec{x} < \vec{z}$ if $\vec{x} \leqslant \vec{z}$ and for some j we have strict inequality: $x_j < z_j$.
- We write $\vec{x} \ll \vec{z}$ if for all j, $x_j < z_j$.
- Throughout the paper, $i \in [n]$, $j \in [m]$. $\vec{x}^i$ denotes the ith column vector of a matrix X.
- Given a matrix $X = (x^i_j) \in \Re^{n\times m}$, $X + \vec{z} = (\vec{x}^1 + \vec{z}, ..., \vec{x}^n + \vec{z})$

2.2 Basic Setup

We study a setup with m infinitely divisible resources, wanted by n agents.

DEFINITION 1. *Given quantities for goods $\vec{q} \in \Re^m_+$, an* allocation *is a matrix $X \in \Re^{n\times m}_+$ with $\sum_i x^i_j \leqslant q_j$ for all j.*

Unless stated otherwise, we assume without loss of generality that $q_j = 1$ for all j, in which case x^i_j denotes the fraction of good j that bidder i gets.

Each bidder i has a utility function $u_i(x^i_1, ..., x^i_m)$ that denotes his utility from receiving the bundle composed of x^i_j fraction of each resource j. Our definitions and results are insensitive to the cardinal properties of u_i and only depend on ordinal ones, so we could have equivalently modeled agent preferences using a preference relation $\prec_i$. Specifically, it means that we do not make any *interpersonal comparisons of utility* [10]. We make the standard assumption that utility functions are non-decreasing, that is - if $\vec{x} \geq \vec{y}$, then $u(\vec{x}) \geqslant u(\vec{y})$. Additionally, each agent has an entitlement e_i, with $\sum_i e_i = 1$, intuitively stating how much he brought into the system or "deserves" of the system. It is useful to think of an agent's entitlement as $1/n$, when n is the number of agents, but, following [6], we also allow any pre-defined entitlements e_i such that $\sum_i e_i = 1$, where the general intention is that an agent with twice the entitlement of another one "deserves" twice the allocation.

DEFINITION 2. *x is called* maximal *for u if for all y, $u(x) \geqslant u(y)$.*

DEFINITION 3. *A utility function u is called* strictly monotonic *if for every non-maximal $\vec{x}$, and all $\vec{y} >> \vec{x}$. We say it is* non-satiable[3] *if it is strictly monotonic and has no maximal $\vec{x}$.*

If u is not *non-satiable*, we say that it is *satiable*, and u is *satiated at $\vec{x}$* if $u(\vec{x}) \geqslant u(\vec{y})$ for all $\vec{y} \in \Re^m_+$.

2.3 Parsimonious Allocations

For the utilities in question, it is often the case that an agent can relinquish some of his bundle without reducing his utility. For clarity, and without losing generality, we will focus on allocations where this is not true. We use $\vec{x} \Downarrow \vec{z}$ to denote a pointwise-minimum, that is $\vec{x} \Downarrow \vec{z} = (min(x_1, z_1), ..., min(x_m, z_m))$.

DEFINITION 4. *A bundle $\vec{x}$ is a* parsimonious bundle *for u if for all $\vec{z} < \vec{x}$ we have that $u(\vec{z}) < u(\vec{x})$. An allocation is called* parsimonious *w.r.t utility functions $u_1, ..., u_n$ if each $\vec{x}^i$ is a parsimonious bundle w.r.t to u_i.*

PROPOSITION 1. *If u is continuous, then for every bundle $\vec{x}$ there exists a parsimonious bundle $\vec{y}$ such that $\vec{y} \leqslant \vec{x}$ and $u(\vec{x}) = u(\vec{y})$.*

[3]One can verify that strictly monotonic and non-satiable is equivalent to the standard economic term "locally non-satiable for all x".

PROOF. Let $L = \{\vec{z} \in \Re^m_+ | u(\vec{x}) = u(\vec{z})\ and\ \vec{z} \leqslant \vec{x}\}$. This set is compact (it is closed by the continuity of u and bounded since $\vec{z} \leqslant \vec{x}$), so a $\vec{y} \in L$ that minimizes $\sum_j y_j$ over all $\vec{z} \in L$ exists. Since $\vec{y} \in L$, we must have $\vec{y} \leqslant \vec{x}$ and $u(\vec{x}) = u(\vec{y})$. For any bundle such that $\vec{z} < \vec{y}$, it holds that $\vec{z} \leqslant \vec{x}$ and $\sum_j z_j < \sum_j y_j$. Hence, it must be the case that $u(\vec{z}) < u(\vec{x})$, otherwise $z \in L$ and $\vec{y}$ does not minimize $\sum_j z_j$ in L . □

2.4 Perfect Complementarity

Perfectly Complementary utility functions could be defined in several ways.

DEFINITION 5. *A utility function u is* perfectly complementary *if for all $\vec{x}, \vec{y}$ we have that $u(\vec{x} \Downarrow \vec{y}) = min\,(u(\vec{x}), u(\vec{y}))$.*

PROPOSITION 2. *The following are equivalent if u is continuous:*

1. *u is perfectly complementary.*
2. *For all parsimonious bundles $\vec{x}, \vec{y}$, either $\vec{x} \leqslant \vec{y}$ or $\vec{y} \leqslant \vec{x}$.*
3. *There exists a function $w : \Re_+ \to (\Re_+ \cup \{\infty\})^m$, called the* parsimonious bundle representation *of u, such that for all $t \geqslant 0$ we have that $u(\vec{x}) \geqslant t$ if and only if $\vec{x} \geqslant w(t)$. That is, u obtains utility level at least t exactly when it gets at least $w_j(t)$ amount of every good j.*

PROOF. Assume u is continuous.

$1 \Rightarrow 2$ ($\neg 2 \Rightarrow \neg 1$): Let $\vec{x}, \vec{y} \in \Re^m_+$ be two parsimonious bundles such that neither $\vec{x} \geqslant \vec{y}$ nor $\vec{y} \geqslant \vec{x}$. Let $\vec{z} = \vec{x} \Downarrow \vec{y}$. This means that there is some j for which $x_j < y_j$ and some l such that $x_l > y_l$. Clearly, $\vec{z} < \vec{x}$ and $\vec{z} < \vec{y}$. Since they are both parsimonious bundles, $u(\vec{z}) < u(\vec{x})$ and $u(\vec{z}) < u(\vec{y})$, and therefore $u(\vec{z}) < min\,(u(\vec{x}), u(\vec{y}))$.

$2 \Rightarrow 3$: Define $w(t)$ for all $t \geqslant 0$ to be a parsimonious bundle achieving utility level t (and ∞ if such a bundle does not exist). We need to show that $u(\vec{x}) \geqslant t$ if and only if $\vec{x} \geqslant w(t)$.

($\Leftarrow$) Take a bundle $\vec{y} \geqslant w(t)$. Then, from the monotonicity of u, $u(\vec{y}) \geqslant u(w(t)) = t$.

($\Rightarrow$) Let $\vec{y}$ be some bundle such that $u(\vec{y}) \geqslant t$ and let $\vec{z} \leqslant \vec{y}$ be a parsimonious bundle with $u(\vec{z}) = u(\vec{y})$ (which exists by Proposition 1). Since $\vec{z}$ and $w(t)$ are both parsimonious, by assumption, either $\vec{z} \geqslant w(t)$ or $\vec{z} \leqslant w(t)$. If it is the latter, $u(\vec{z}) \leqslant u(w(t)) = t$ which means $u(\vec{z}) = t$ resulting in $\vec{z} = w(t)$ (otherwise, $w(t)$ is not parsimonious). Either way, $\vec{y} \geqslant \vec{z} \geqslant w(t)$, as required.

$3 \Rightarrow 1$: We trivially have that $u(\vec{x} \Downarrow \vec{y}) \leqslant min\,(u(\vec{x}), u(\vec{y}))$ so we only need to show the opposite inequality. Let $t = min(u(\vec{x}), u(\vec{y}))$. From (3) it follows that $\vec{x} \geqslant w(t)$ and $\vec{y} \geqslant w(t)$, and so $\vec{x} \Downarrow \vec{y} \geqslant w(t)$ so $u(\vec{x} \Downarrow \vec{y}) \geqslant t$ as required.

This concludes the proof that the three definitions are equivalent for a continuous utility function u. □

LEMMA 1. *If $\vec{x}$ is a parsimonious bundle and u is a perfectly complementary utility function, then $\vec{x} = w(u(\vec{x}))$*

PROOF. Since u is perfectly complementary, by Proposition 2, $\vec{x} \geqslant w(u(\vec{x}))$. Since $\vec{x}$ is a parsimonious bundle, for all $\vec{z} < \vec{x}$, $u(\vec{z}) < u(\vec{x})$. Let $\vec{z} = w(u(\vec{x}))$. By definition, $u(\vec{z}) = u(\vec{x})$. However, if there exists some j such that $x_j > z_j$, it contradicts $\vec{x}$ being parsimonious. □

Note that Proposition 2 implies that $w(t)$ must be non-decreasing. However, the parsimonious bundle representation w of a continuous perfectly complementary utility u is not always continuous itself. It turns out that u being strictly monotonic is equivalent to the continuity of w (for all $t \in [0, max_{\vec{x}} u(\vec{x})]$).

LEMMA 2. *A continuous perfectly complementary utility u is strictly monotonic if and only if its parsimonious bundle representation w is continuous on the domain $\{t|0 \leqslant t \leqslant max_{\vec{x}}u(\vec{x})\}$.*

PROOF. (If): Take a point $\vec{x}$ for which u is not satiated, $u(\vec{x}) = t_x$, and some $\vec{y}$ such that $\vec{y} \gg \vec{x}$. Consider $w(t_x)+\varepsilon)$ for some $\varepsilon > 0$, which is a parsimonious bundle with utility $t_x+\varepsilon > t_x$. (Since u is not satiated at $\vec{x}$, $\infty \gg w(t_x)+\varepsilon)$ for a small enough ε). Since $\vec{y} \gg \vec{x} = w(t_x)$ (by Lemma 1), by continuity of w, for sufficiently small ε we still have $\vec{y} \gg w(t_x)+\varepsilon)$, and so $u(\vec{y}) \geqslant t_x+\varepsilon > t_x$, as needed.

(Only if): Let (α, z_{-j}) be the vector obtained from replacing $\vec{z}$'s jth coordinate by α. By definition, each coordinate of w is non-decreasing, so to prove continuity it suffices to prove for every coordinate j that there are no gaps in the values that $w_j(t)$ attains. Assume that a certain value $\beta = w_j(t)$ is achieved. We need to show that for all $0 \leqslant \alpha < \beta$, there exists a parsimonious bundle $w(t')$ such that $w_j(t') = \alpha$.

Take some bundle $\vec{z} \gg w(t)$. Let $t' = u(\alpha, z_{-j})$, and consider $w(t')$. From the monotonicity of u, and Proposition 1, it follows that $\vec{z} \gg w(t) \geqslant w(t')$. Since $w(t')$ is parsimonious, by definition, $w_j(t') \leq \alpha$. It must be, therefore, that $w_j(t') = \alpha$, since otherwise, $w(t') \ll (\alpha, z_{-j})$, which contradicts u being strictly monotonic. □

In addition, perfectly complementary utilities are inherently quasi-concave.

DEFINITION 6. *u is* quasi-concave *if $u(\vec{x}) \geqslant u(\vec{y})$ implies that $u(\lambda\vec{x}+(1-\lambda)\vec{y}) \geqslant u(\vec{y})$ for all $0 < \lambda < 1$. It is* strictly quasi-concave *if $u(\vec{x}) > u(\vec{y})$ implies that $u(\lambda\vec{x}+(1-\lambda)\vec{y}) > u(\vec{y})$ for all $0 < \lambda < 1$.*

LEMMA 3. *Every perfectly complementary utility u is quasi-concave.*

Proof is omitted, as it is similar to that of lemma 4.

If the utility function u is also strictly monotonic, we can make the following stronger statement:

LEMMA 4. *Every perfectly complementary and strictly monotonic utility function u is strictly quasi-concave.*

PROOF. Let $\vec{x}, \vec{y}$ be two bundles such that $u(\vec{x}) > u(\vec{y})$. Let $u(\vec{x}) = t_1$, $u(\vec{y}) = t_2$, and $\vec{z} = \lambda\vec{x}+(1-\lambda)\vec{y}$. Given j, if $y_j \geqslant x_j$, then $y_j \geqslant z_j \geqslant x_j \geqslant w_j(t_1)$. Otherwise, $x_j > z_j > y_j$, there is some $\hat{t}_j > t_2$ such that $w_j(\hat{t}_j) = z_j$ (Lemma 2). For $\hat{t} = min(min_j(\hat{t}_j), t_1)$, $\hat{t} > t_2$ and therefore $\vec{z} \geqslant w(\hat{t})$. Hence, by definition $u(\vec{z}) \geqslant \hat{t} > t_2 = u(\vec{y})$ as required. □

Example: Leontief Functions

Leontief utilities are one example of perfectly complementary utility functions. These utility functions express an interest in a certain fixed *proportion of resources*.

u is *Leontief* if $u(\vec{x}) = \min_{j\in S}(x_j/r_j)$ for some constants $\vec{r} \geqslant 0$, where $S = \{j|r_j > 0\}$. This means that in order to obtain utility level $t \geqslant 0$, the agent needs at least $r_j t$ fraction of each good j. Thus, the parsimonious bundle for each utility level t is $w(t) = (r_1 t, ..., r_m t)$.

Leontief utilities are non-satiable. The utilities considered in [6] are satiable versions of Leontief utilities:
$u(\vec{x}) = \min(1, \min_{j\in S}(x_j/r_j))$, where $0 \leqslant r_j \leqslant 1$; i.e. the maximum utility of 1 is obtained at the minimal saturation bundle $(r_1...r_m)$. We call these *satiable Leontief utilities*.

Note that a parsimonious bundle $\vec{x}$ for an agent with a Leontief utility needs to obtain the minimum on all coordinates, and therefore maintains $\vec{x} = \alpha\vec{r}$ for some constant α.

2.5 Efficiency

The first requirement from an allocation is obviously to be efficient. We start with a very weak notion of efficiency that intuitively does not assume the possibility of trade. Though in general this notion of efficiency is significantly weaker than Pareto efficiency, we show that for perfectly complementary utilities it implies Pareto efficiency.

DEFINITION 7. *An allocation is* non-wasteful *if :*

1. *For all i, the bundle $\vec{x}^i$ is* parsimonious *for u_i.*

2. *$\forall i$ $u_i(\vec{x}^i + \vec{z}) = u_i(\vec{x}^i)$ where $z_j = 1 - \sum_i x^i_j$.*

Intuitively, these are allocations where no agent gets more unless it improves his utility, and goods are left on the table only if they cannot improve the agents' utilities.

Our interest is in economies where trade does not promote efficiency (the opposite of what is usual in economic theory).

Let $\vec{u}(X) = (u_1(\vec{x}^1), ..., u_n(\vec{x}^n))$.

DEFINITION 8. *An allocation X is* Pareto efficient *if there is no other allocation Z such that $\vec{u}(Z) > \vec{u}(X)$.*

Pareto-efficiency is stronger than being non-wasteful as it also requires that no bilateral (and multi-lateral) trade that is mutually beneficial is possible, while non-wasteful only requires goods not to be left on the table if they can be of any use to anyone. However, for perfectly complementary utilities, non-wastefulness is a sufficient condition for Pareto-efficiency. To obtain equivalence we must add the requirement of parsimony to Pareto-efficiency. We can add it without loss of generality as we can replace any Pareto efficient allocation X by a parsimonious allocation $Y \leqslant X$ that provides all agents with the same utilities.

DEFINITION 9. *An economy $u_1, ..., u_n$ is called a no-trade economy if every non-wasteful allocation is also Pareto efficient.*

PROPOSITION 3. *An economy composed of perfectly complementary utilities is a no-trade economy.*

PROOF. Let X be some non-wasteful allocation for n perfectly complementary bidders. Let $z_j = 1 - \sum_i x^i_j$. Since X is non-wasteful, for all i, $u_i(\vec{x}^i) = u_i(\vec{x}^i + \vec{z})$. Assume, by way of contradiction, that there is an allocation Y such that $\vec{u}(Y) > \vec{u}(X) = \vec{u}(X+\vec{z})$. Therefore, there is some i for which $u_i(\vec{y}^i) > u_i(\vec{x^i})$. We can assume, w.l.o.g, that Y is a parsimonious allocation. By Proposition 2, $\vec{y}^i > \vec{x}^i$. Since Y is an allocation, it must hold that $y^i - x^i \leqslant 1 - \sum_k x^k$, but that contradicts X being non-wasteful. □

3. FAIRNESS NOTIONS

Fairness of an allocation is an elusive concept. There are many opinions on what counts as fair, which sometimes vary depending on the application. As a result, there are many fairness notions for allocations in various models. Some recent papers dealing with the same motivation suggested different fairness notions. One was introduced by Dolev et al. [6] and the other by Ghodsi et al. [9]. In this section, we rephrase the definitions of [6] and [9], generalize, and compare them.[4]

[4]In the case of Ghodsi et al., since the paper has a systems flavor, we use our understanding of their somewhat informal definitions.

3.1 Dominant / Generalized Resource Fairness

The notion of "Dominant Resource Fairness" was advocated in [9]. Conceptually, we view this notion as combining two elements: the first is the choice of defining fairness between the agents in the Rawlsian [12] sense of maximizing the welfare of the poorest agent, and the second is the choice of quantifying the allocation of an agent according to the maximum share of any resource he got. We generalize this notion of fairness by sticking to the first element of fairness and allowing a variety of choices for the second.

Let $||\cdot||$ be a norm on $\Re^m$. We think of $||\vec{x}^i||$ as our measure of "how much" i obtains - a scalar that quantifies the bundle $\vec{x}^i$. In fact, we don't really need all the properties of a norm, just monotonicity and continuity.

DEFINITION 10. *Given a norm $||\cdot||$ on $\Re^m$ and two parsimonious allocations $X, Y \in \Re_+^{n\times m}$ (where $\vec{x}^i, \vec{y}^i$ are column vectors in X, Y respectively). We say that X is fairer than Y (according to $||\cdot||$) if*

$$\min_{i:||\vec{x}^i||\neq||\vec{y}^i||}(||\vec{x}^i||) \geqslant \min_{i:||\vec{x}^i||\neq||\vec{y}^i||}(||\vec{y}^i||)$$

That is, if we look at the agent with the minimal allocation that differs between X and Y, then that agent gets (weakly) more in X. Generalizing to the case that agents come with pre-defined entitlements, we also say that X is fairer than Y under entitlements $e_1, ..., e_n$ if

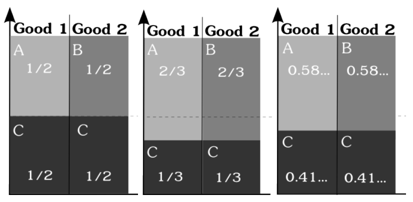
$$\min_{i:||\vec{x}^i||\neq||\vec{y}^i||}(||\vec{x}^i/e_i||) \geqslant \min_{i:||\vec{x}^i||\neq||\vec{y}^i||}(||\vec{y}^i/e_i||)$$

We say that an allocation X is $||\cdot||$-fair *if for every other allocation Y, we have that X is fairer than Y (And similarly for fairness under entitlements.).*

In [9], the L_∞ norm was used, and the fairness notion was termed "Dominant Resource Fairness" (DRF). Other natural choices would be the L_1 norm, referred to as "Asset Fairness" in [9], which counts total resource use, and intermediate norms such as L_2. The following example shows the different fair allocations for these three choices of norm:

Agents	e	Goods 1	2
A	1/3	1	0
B	1/3	0	1
C	1/3	1	1

(a) Request Matrix



(b) L_∞ **(c)** L_1 **(d)** L_2

Figure 3: Figure 3a depicts the agents' interest in the resources and their respective entitlements. Figures (b)-(d) show the different allocations corresponding to three different norms. $0.41..$ is the solution to the equation $2x^2 = (1-x)^2$.

As seen in Figure 3, it is hard to tell which norm provides an intuitively fairer allocation, and each may be appropriate in some cases. One may argue that L_∞ is unfair since agent C receives too much in total resources, while L_1 is unfair as C receives too little of any specific resource. L_2 gives an intermediate allocation, but then again, it may be argued that C still receives too much in total.

3.2 No Justified Complaints

This fairness notion was originally stated using Leontief utilities. We suggest a generalization of this fairness notion to perfectly complementary utilities:

DEFINITION 11. **No Justified Complaints for PC utilities** *An allocation X has the property of* "No Justified Complaints" *(NJC) if for all agents i:*

1. *$\vec{x}^i$ is a parsimonious bundle.*
2. *Either there is some "bottleneck" good j such that $\sum_i x_j^i = 1$ with $x_j^i \geqslant e_i$, or u_i is satiated at $\vec{x}^i$: $u_i(\vec{x}^i) \geqslant u_i(\vec{z})$ for all $\vec{z} \in \Re^m$.*

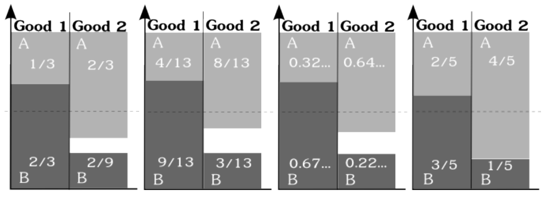
If an allocation has the NJC property, we will say it is *"Bottleneck-Based Fair"* (BBF).

Restricting the above definition to satiable Leontief utilities reduces to the original definition in [6].

One can easily verify that the examples of $||\cdot||$-fair allocations above are all BBF. Of course, all criticism of these allocations applies to BBF as well. However, not all $||\cdot||$-fair allocations are BBF. The examples in Figure 4 show that BBF and norm-fairness are unrelated notions.

Agents	e	Goods 1	2
A	1/2	1/2	1
B	1/2	1	1/3

(a) Request Matrix



(b) L_∞ **(c)** L_1 **(d)** L_2 **(e)** BBF

Figure 4: In the setup above, none of the $||\cdot||$-Fair allocations is BBF, since agent A does not have his entitlement of a bottleneck resource. (4e) shows a BBF allocation of this setup, that does not satisfy any of the aforementioned norm-fairness notions.

4. COMPUTING A NORM-FAIR ALLOCATION

The basic idea of [9] is that a greedy allocation rule approaches dominant resource fairness: at each stage the "poorest" agent gets another ε of his required bundle. We observe that the same is true in general for all $||\cdot||$-fairness notions, and show how the pseudo-polynomial algorithm presented in [9] may be converted into a fully polynomial-time allocation algorithm. We will present the algorithm in general terms and assume the following minimal type of oracle access to the underlying utilities and norm.

Our algorithm applies to continuous, strictly monotonic, perfectly complementary utilities with the following additional property:

DEFINITION 12. *Let u be a perfectly complementary function, and $\vec{x}, \vec{y} \in \Re_+^m$ be two parsimonious bundles with respect to u. We call u* compatible with a norm $||\cdot||$ *if $u(\vec{x}) > u(\vec{y})$ implies $||\vec{x}|| > ||\vec{y}||$.*

LEMMA 5. *Every perfectly complementary utility is compatible with L_p for every $1 \leqslant p < \infty$. Leontief and satiable Leontief utilities are also compatible with L_∞.*

PROOF. Let u be a perfectly complementary utility function, and let $\vec{x}, \vec{y}$ be two parsimonious bundles such that $u(\vec{x}) > u(\vec{y})$. Let $u(\vec{x}) = t$ and $u(\vec{y}) = t'$. Since $\vec{x}, \vec{y}$ are parsimonious, and u is monotonic, it follows that $\vec{x} \geq \vec{y}$ (Proposition 2). It cannot be the case that $\vec{x} = \vec{y}$ since $t > t'$, so $\vec{x} > \vec{y}$.

$\vec{x} > \vec{y}$, so $x_j \geqslant y_j$ for all j, with at least one inequality strict. Therefore, $\sum_j {x_j}^p > \sum_j {y_j}^p$ and from the monotonicity of $\sqrt[p]{}$, we obtain that $||\vec{x}||_p > ||\vec{y}||_p$ as required.

For Leontief utilities, these parsimonious bundles are $t \cdot \vec{r}$ and $t' \cdot \vec{r}$, so in the case of L_∞: $||t \cdot \vec{r}||_\infty = t \cdot max_j(\vec{r}) > t' \cdot max_j(\vec{r}) = ||t' \cdot \vec{r}||_\infty$, as required.

The above applies trivially to satiable utilities as well, since for bundles at which u is satiated, there will not exist a bundle that gives a higher utility. □

Lemma 5 implies that if a perfectly complementary utility function is compatible with a norm, every $|| \cdot ||$-fair value corresponds to a single parsimonious allocation.

We need to describe how a utility function u_i and a norm $|| \cdot ||$ are accessed. We assume that is provided by the following type of oracle query:

Given a fairness level h, return the parsimonious bundle $\vec{x}^i$ for i with $||\vec{x}^i|| = h$.

Notice that this type of oracle access is no stronger than direct access to u_i or its corresponding parsimonious representation w^i, as given access to $w^i(t)$ one can use binary search over t to find a bundle with $||w^i(t)|| = h$, and given only direct access to u_i one can recover $w^i(t)$ by binary search on every index (i.e., $w^i_1(t)$ is the smallest α such that $u_i(\alpha, 1, ..., 1) = t$). We prefer this formalization of access since it does not directly use the cardinal utilities but only their ordinal properties, and so will often be more natural. For example, for Leontief utilities, where $u_i(\vec{x}) = \min_j(x_j/r_j)$, the reply for query h would be $\vec{x} = h \cdot \vec{r}/||\vec{r}||$.

We use this oracle in the following basic step of the algorithm:

Allocation Step(S,q) (For a given $|| \cdot ||$) :
Input:
A set S of agents and a vector of remaining quantities $q = (q_1 ... q_m)$.
Output: Returns the maximum h such that each $i \in S$ can be given a parsimonious $\vec{x}^i$ with $||\vec{x}^i|| = h \cdot e_i$ and yet stay within the given quantities: for all j, $\sum_i x^i_j \leqslant q_j$. It also returns the parsimonious allocation (X) with $||\vec{x}^i|| = h$.
Algorithm: To find h, one may use binary search given the access to the utility functions specified above.

Note that this algorithm allows us to get arbitrarily close to h, but cannot guarantee that h is accurate. This step is therefore, in general, pseudo-polynomial.

For the special case of Leontief utilities where for each agent i, $u_i(\vec{x}) = \min_j(x_j/r^i_j)$, the following is a direct solution: $h = \min_j(q_j \cdot ||\vec{r}^i|| / \sum_i e_i r^i_j)$.

There are two possible reasons that may limit h: either one of the agents gets satiated at the parsimonious bundle corresponding to h or (at least) one of the items got exhausted.

LEMMA 6. *If all utilities are perfectly complementary, continuous, strictly monotonic and compatible with $|| \cdot ||$ then Allocation-Step returns an allocation where either for some $i \in S$, $u_i(\vec{x}^i) \geqslant u_i(\vec{y})$ for all $\vec{y}$ or for some j for which $q_j > 0$, $\sum_i x^i_j = q_j$.*

PROOF. Let $t_i = u_i(\vec{x}^i)$. Assume by way of contradiction that there is some $\varepsilon > 0$ small enough such that for all j, $\sum_{i \in S} w^i_j(t_i + \varepsilon) \leqslant q_j$ (note that if t_i is the maximal value of u_i, then $w^i_j(t_i + \varepsilon) = \infty$ for all j). By Lemma 2, it must be the case that $w^i(t_i + \varepsilon) > w^i(t_i)$. Since u_i is compatible with $|| \cdot ||$ for all i, $||w^i(t_i + \varepsilon)|| > ||w^i(t_i)|| = h$ for all i, which contradicts h's maximality under the constraints. □

Now we have the following algorithm:

```
Initialize $S$ to be all agents and $q$ to be initial quantities.
while $S \neq \emptyset$ do
    $h, (X) =$ AllocationStep$(S, q)$.
    $G \leftarrow \{j | \sum_i x^i_j = q_j\}$ // Exhausted items
    Let $F$ be the set of agents $i$ such that for all parsimonious
    bundles $\vec{y}$ with $||\vec{y}|| > h$ we have $y_j > x^i_j$ for some $j \in G$.
    $q = q - \sum_{i \in F} \vec{x}^i$
    $S = S \setminus F$
end while
```

THEOREM 1. *This algorithm outputs the $|| \cdot ||$-fair allocation under entitlements $e_1, ..., e_n$ for all continuous, strictly monotonic and perfectly complementary utility functions compatible with $|| \cdot ||$.*

PROOF. First we notice that this algorithm terminates, since at every allocation step either some agent is satiated at the parsimonious allocation corresponding to h (and then leaves S) or some good is exhausted (and enters G), and since S never increases and G never decreases, we can have at most $n + m$ allocation steps.

We now prove correctness, by induction over the number of iterations of the main loop. Let Y be a parsimonious $|| \cdot ||$-fair allocation, and X be the allocation of the current step of the algorithm, and let X' be the allocation of the previous step. Similarly, let S' be the set of non-exhausted agents in the end of the previous step, and S be the set at the end of the current step. Note that for all i, $\vec{x}^i \geqslant \vec{x}'^i$, since the algorithm never diminishes allocations. We prove the following invariants by induction. The theorem follows directly.

1. If $i \in S$ then $||\vec{x}^i|| \leqslant ||\vec{y}^i||$.
2. If $i \notin S$ then $||\vec{x}^i|| = ||\vec{y}^i||$.

Basis: Right after the initialization phase, $\vec{x}^i = \vec{0}$ for all i (which means $||\vec{x}^i|| = 0$) and $S = [n]$) so (1) holds trivially and (2) holds vacuously.

Step: Assume that the above is true for X'. For all $i \notin S'$ $||\vec{y}^i|| = ||\vec{x}'^i|| = ||\vec{x}^i||$, from the assumption and the fact that the algorithm does not change allocations of agents not in S'. Since we assumed Y is $||\cdot||$-fair, for all $i \in S'$, $||\vec{y}^i|| \geq ||\vec{x}^i|| = h$ since otherwise $\min_{\{i:||\vec{x}^i|| \neq ||\vec{y}^i||\}} ||\vec{x}^i|| > \min_{\{i:||\vec{x}^i|| \neq ||\vec{y}^i||\}} ||\vec{y}^i||$ and Y's fairness is contradicted. Since $S \subsetneq S'$, (1) is proven. Now, let's look at $i \in S' \setminus S$. It can't be that $||\vec{y}^i|| > h$ since AllocationStep chooses the maximal h for which the allocation is both parsimonious and valid, so $||\vec{y}^i|| = h$ for each $i \in S' \setminus S$. We obtain therefore that for all $i \notin S$, $||\vec{y}^i|| = ||\vec{x}^i||$ as required.

Observe that these two invariants imply that at the end of the algorithm, $X = Y$ for any $|| \cdot ||$-fair allocation, which not only proves the correctness of the algorithm, but also the uniqueness of the fair allocation. □

5. COMPETITIVE EQUILIBRIA AND BOTTLENECK BASED FAIRNESS

In this section we develop a close connection between the problem of finding a BBF allocation and the problem of finding an equilibrium in a Fisher Market, a special case of an Arrow-Debreu market that is often more computationally tractable. As corollaries of this connection and known results, we get both a general existence result, generalizing that of [6], and a polynomial-time algorithm for agents with Leontief utilities.

A Fisher Market is a market model where n unrelated agents are interested in m different, infinitely divisible goods, each available in some quantity q_j. Each agent has a preference over bundles, given by a utility function u_i, and a budget ("money") that he will use to trade with the other agents towards a better bundle. The question of the existence and computation of equilibria in this setup has been well studied in mathematical economics, and more recently in computational economics (to state just a few references: [1, 7, 8], see [4] for a recent survey).

5.1 Fisher Market Equilibrium: Definition and Existence

A market equilibrium is a state where all agents are no longer interested in trading with their peers. In a Fisher Market, an equilibrium state is defined by the following two conditions (see [4]).

DEFINITION 13. *In a market setting with n agents interested in m infinitely divisible goods, where agents' preferences over bundles are given by a utility function u_i, and each agent has a budget $e_i > 0$, an equilibrium is defined as a pair of price vector $\pi \in \Re_+^m$ and an allocation $X \in \Re_+^{n\times m}$ with the following two properties:*

1. *The vector $\vec{x}^i$ maximizes $u_i(\vec{x}^i)$ under the constraints $\sum_j \pi_j x_j^i \leqslant e_i$ and $\vec{x}^i \in \Re_+^m$.*

2. *For each good $j \in [m]$, $\sum_i x_j^i = q_j$*

It is imperative to understand that the two conditions are in some sense independent. When an agent calculates the bundle satisfying condition 1, he **does not** take into account the availability of goods in the market. If agents demand more than the available goods, the prices must go up - until an equilibrium is reached.

Fisher Market is a special case of the market studied by Arrow and Debreu, where all agents arrive to the market with the same proportions of all goods ([4]). Thus, as a corollary of the Arrow and Debreu Theorem ([1]), given a Fisher market as described above, an equilibrium exists if the following conditions hold:

- u_i is continuous
- u_i is strictly quasi-concave (Definition 6).
- u_i is non-satiable:[5] For all $\vec{x} \in \Re_+^m$, there is an $\vec{x}' \in \Re_+^m$ such that $u_i(\vec{x}') > u_i(\vec{x})$

From Lemma 4, every perfectly complementary and strictly monotonic utility function is strictly quasi-concave, so we obtain the following corollary:

COROLLARY 1. *For a setup with continuous, strictly monotonic, non-satiable and perfectly complementary utility functions u_i, there exists a Fisher market equilibrium.*

[5]Note that if u is strictly monotonic (Definition 3) and there is no $\vec{x}$ in which u is satiated, then it is non-satiable.

5.2 The Non-Satiable Case

The main point of this subsection is that market equilibrium for continuous perfectly complementary and *non-satiable* utilities is BBF. The next subsection will observe that the non-satiability is not really required for strictly monotonic utilities.

THEOREM 2. *Let $u_1, ..., u_n$ be perfectly complementary, continuous, strictly monotonic and non-satiable utility functions. Let $e_1, ..., e_n$ be the agents' budgets, let (π, X) be a Fisher Market equilibrium in a market where $q_j = 1$ for all j, and let $Y \leqslant X$ be the parsimonious allocation such that for all i, $u_i(\vec{y}^i) = u_i(\vec{x}^i)$. Then Y is BBF.*

PROOF. Let $F = \{j | \pi_j = 0\}$ be the "free" goods. First notice that since the u_is are strictly monotonic, continuous and non-satiable, we must have that all agents exhaust their budget, that is $\sum_j \pi_j x_j^i = e_i$ for all i, as otherwise the remaining money could then be used to buy a little bit extra of each good, thereby increasing i's utility, contradicting the requirement that $\vec{x}^i$ maximizes i's utility within budget. As $\sum_i x_j^i = q_j = 1$, it follows that $\sum_j \pi_j = \sum_j \sum_i x_j^i \pi_j = \sum_i \sum_j x_j^i \pi_j = \sum_i e_i = 1$. Now notice that for $j \notin F$ we must have $x_j^i = y_j^i$ for all i, since otherwise agent i could demand only y_j^i of good j saving $(x_j^i - y_j^i)\pi_j$ money, which could then be used to buy a little bit extra of each good. Thus, $e_i = \sum_j \pi_j x_j^i = \sum_{j \notin F} \pi_j x_j^i = \sum_{j \notin F} \pi_j y_j^i$. But since $\sum_{j \notin F} \pi_j = \sum_j \pi_j = 1$ we have a convex combination of y_j^i over all $j \notin F$ that sums to e_i, and it follows that for some $j \notin F$, $y_j^i \geqslant e_i$. The proof is concluded by noting that all $j \notin F$ are bottleneck resources in the allocation Y, since for all $j \notin F$, $1 = \sum_i x_j^i = \sum_i y_j^i$. $\square$

5.3 The Satiable Case

Theorem 2 applies to almost all continuous, strictly monotonic and perfectly complementary utilities. The crux lies in the non-satiability assumption. In this subsection, we show that this assumption is, in fact, without loss of generality. Starting with a satiable, strictly monotonic, continuous and perfectly complementary function, we can convert it to a similarly characterized non-satiable function by adding a resource that only the agent whose utility is satiable is interested in.

Given a parsimonious bundle representation w^i of a strictly monotonic, perfectly complementary and satiable utility function u_i, and assuming, w.l.o.g, that the maximal value of u_i is 1 (otherwise, we normalize the function, without affecting any of its other properties), we define the parsimonious representation w'^i of a continuous, perfectly complementary, strictly monotonic and non-satiable utility as follows:

$$w'^i(t) = \begin{cases} \left(w_1^i(t), ..., w_m^i(t), t\right) & t \leqslant 1 \\ t \cdot \left(w_1^i(1), ..., w_m^i(1), 1\right) & t > 1 \end{cases}$$

Let u_i' be the perfectly complementary utility function defined by w'^i.

PROPOSITION 4. *If u_i is continuous, strictly monotonic, and perfectly complementary, then u_i' is continuous, strictly monotonic, perfectly complementary and non-satiable. Moreover, if u_i is satiable Leontief, then u_i' is Leontief.*

PROOF. Since we defined u_i' using the parsimonious bundle representation w^i, it is perfectly complementary by definition. It's also trivial to see that u is non-satiable. It remains to show it is continuous and strictly monotonic. Both are true for $t < 1$ as witu_i'

coincides with u_i there; both are true for $t > 1$ as u'_i is just a Leontief utility in that range; and at $t = 1$ we get the same value from the two parts, thereby maintaining both continuity and consistency.

If u_i is satiable Leontief, then u'_i could be otherwise written as: $u''_i = \min_{j\in[m+1]}(x^i_j/r^i_j)$, where $r^i_{m+1} = 1$, and the remaining constants are unchanged. This is clearly a standard Leontief function, and it is trivially equivalent to the function constructed above. □

LEMMA 7. *Let u'_i be the extension of the satiable perfectly complementary utility u_i for all i. Then if an allocation is BBF with respect to the u'_is it is also BBF with respect to the u_is.*

PROOF. Given an allocation $X = (\vec{x}^1, ..., \vec{x}^n)$ that is an equilibrium with respect to $(u'_1, ..., u'_n)$, we need to show that the conditions of BBF hold with respect to $(u_1, ..., u_n)$. We know that for each agent i, there exists some $j \in J$ such that $x^i_j \geqslant e_i$, where $J = \{j | \sum_i x^i_j = 1\}$ (since u'_i has no maximum, the other option is irrelevant).

To show that with respect to the original utilities and setup the very same allocation is BBF, we have to discard all "virtual resources" added when extending the utilities.

Let $J = \{j | \sum_i x^i_j = 1\}$ be the set of bottleneck resources in the given allocation. Let $J' \subseteq J$ be the set of bottleneck resources after the virtual ones have been discarded. Given an agent i, if J does not include the "virtual resource?âĆňâĎć?âĆňâĎć added for i, then after the discard there still exists $j \in J'$ such that $x^i_j \geqslant e_i$ as required. If J does include i's "virtual resource?âĆňâĎć?âĆňâĎć, then it is a bottleneck. This means that agent i got everything available of that resource and therefore, by construction of the utility function u'_i, $x^i_j = w^i_j(\max(u_i(\vec{x})))$ for all $j \in J$ (meaning, agent i got all he asked for). □

5.4 The Bottom Line

Looking at what we have constructed so far, we now have the tools to generalize [6].

THEOREM 3. *In every setup with perfectly complementary, continuous, and strictly monotonic utility functions $u_1, ..., u_n$ there exists an allocation that satisfies "No Justified Complaints".*

PROOF. If needed, we extend the u_is to be non-satiable as shown in the previous section. Then take a competitive equilibrium of the u'_is, guaranteed to exist by Corollary 1, and then by Theorem 2 the parsimonious bundles derived from that allocation are BBF for the u'_is, and thus by Lemma 7 also for the u_is. □

THEOREM 4. *There exists a polynomial-time algorithm that computes a* "Bottleneck-Based Fair *allocation for every setup with Leontief or satiable Leontief utilities.*

PROOF. Codenotti and Varadarajan [3] introduces an algorithm based on convex programming that finds a competitive equilibrium in a Fisher Market for Leontief utilities. In fact, the solution to their convex program provides the relevant parsimonious bundles. For the case of satiable Leontief utilities, the process of converting them to Leontief utilities described in the previous section is algorithmically trivial. Thus, we obtain a polynomial-time algorithm that finds a BBF allocation in every setup with Leontief or satiable Leontief utilities. □

6. DISCUSSION

As computer systems become larger and more pervasive, servicing more and more agents, the question of fair resource allocation becomes more relevant. The classic requirements of an operating system's scheduler will no longer do.

We have studied two notions of fairness: DRF [9] and BBF [6]. The fact that both used the same family of utility function seems to indicate something about the nature of utility functions in such systems. Therefore, we have extended it to a larger family of utility functions - which may apply to other scenarios as well. The generalization of DRF to GRF stresses how vague the concept of fairness is, as it requires a few independent decisions, none of which comes with a strict sense of what is right and what is wrong. Establishing the connection between the question of BBF and Fisher Market equilibrium shows that answers can sometimes be found in unexpected places. It also implies that any solution to the market equilibrium problem for continuous, strictly monotonic and perfectly complementary utilities automatically produces a BBF allocation (for example, [8]). An important issue for further research is the question of incentive. For example, [9] have shown that their algorithm for DRF is strategy proof and envy-free. In addition, the polynomial-time algorithm suggested here for BBF may allow us to prove or disprove the existence of similar properties for this fairness notion.

7. REFERENCES

[1] K. J. Arrow and G. Debreu. The existence of an equilibrium for a competitive economy. *Econometrica*, XXII:265–90, 1954.

[2] D.P. Bertsekas and R.G. Gallager. *Data networks:*. Prentice-Hall, 1987.

[3] Bruno Codenotti and Kasturi R. Varadarajan. Efficient computation of equilibrium prices for markets with leontief utilities. In *ICALP*, pages 371–382, 2004.

[4] Bruno Codenotti and Kasturi R. Varadarajan. Computation of market equilibria by convex programming. In Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani, editors, *Algorithmic Game Theory*. Cambridge University Press, 2007.

[5] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991.

[6] Danny Dolev, Dror G. Feitelson, Joseph Y. Halpern, Raz Kupferman, and Nati Linial. No justified complaints: On fair sharing of multiple resources. In *ITCS*, 2012.

[7] E. Eisenberg. Aggregation of utility functions. *Management Sciences*, 7(4):337–350, 1961.

[8] Dinesh Garg, Kamal Jain, Kunal Talwar, and Vijay V. Vazirani. A primal-dual algorithm for computing fisher equilibrium in the absence of gross substitutability property. *Theor. Comput. Sci.*, 378:143–152, June 2007.

[9] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. Dominant resource fairness: fair allocation of multiple resource types. In *Proceeding NSDI*, 2011.

[10] Peter J. Hammond. Interpersonal comparisons of utility: Why and how they are and should be made. In *Interpersonal Comparisons of Well-Being*, pages 200–254. University Press, 1991.

[11] Jin Li and Jingyi Xue. Egalitarian division under leontief preferences.

[12] John Rawls. *A Theory of Justice*. Cambridge, Massachusetts: Belknap Press of Harvard University Press, 1971.

Session 5D
Auction & Mechanism Design

Mixed-bundling auctions with reserve prices[*]

Pingzhong Tang
Computer Science Department
Carnegie Mellon University
kenshin@cs.cmu.edu

Tuomas Sandholm
Computer Science Department
Carnegie Mellon University
sandholm@cs.cmu.edu

ABSTRACT

Revenue maximization in multi-item settings is notoriously elusive. This paper studies a class of two-item auctions which we call a *mixed-bundling auction with reserve prices (MBARP)*. It calls VCG on an enlarged set of agents by adding the seller—who has reserve valuations for each bundle of items—and a fake agent who receives nothing nor has valuations for any item or bundle, but has a valuation for pure bundling allocations, i.e., allocations where the two items are allocated to a single agent. This is a strict subclass of several known classes of auctions, including the affine maximizer auction (AMA), λ-aution, and the virtual valuations combinatorial auction (VVCA). As we show, a striking feature of MBARP is that its revenue can be represented in a simple closed form as a function of the parameters. Thus, we can solve first-order conditions on the parameters and obtain the optimal MBARP. The optimal MBARP yields significantly higher revenue than prior auctions for which the revenue-maximizing parameters could be solved for in closed form: separate Myerson auctions, pure-bundling Myerson auction, VCG, and mixed-bundling auction without reserve prices. Its revenue even exceeds that obtained via simulation within broader classes: VVCA and AMA.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence — Multi agent system; J.4 [**Social and Behavior Science**]: Economics

General Terms

Theory, Economics

Keywords

Auction, optimal auction, combinatorial auction, revenue maximization, bundling, reserve price

1. INTRODUCTION

Perhaps one of the most important open problem in combinatorial auctions (CAs), and mechanism design at large, is to design revenue-maximizing (aka optimal) auctions. Specifically, the problem is, for the seller, to design an auction that maximizes her expected revenue, subjected to the incentive compatibility (IC) and individual rationality (IR) constraints, given the information about bidders' valuation distributions but not the actual values. There has been a significant amount of research on this topic, but even the 2-item case with additive valuations is open. In fact, the problem of designing an optimal CAs (even in a one-bidder setting) is NP-hard [3], so a general concise characterization cannot exist (unless P=NP). This is in contrast to the one-item setting, where the problem was elegantly solved, by Myerson [11]. This was later generalized to multiple identical units of one item [10].

In this paper, we consider a setting with two items, where each bidder's valuation functions are additive, that is, a bidder's valuation for the bundle of the two items is the sum of his valuations for the individual items. Thus, each bidder has a two-dimensional type: he has a valuation for the first item and a valuation for the second item. We consider the revenue optimization problem within a general symmetric class of auction: *mixed-bundling auction with reserve price (MBARP)*. It is a subclass of existing classes of auctions such as *affine maximizer auctions (AMAs)* [14], *λ-autions* [6], and *virtual valuations combinatorial auctions (VVCAs)* [8, 9, 15]. The dominant-strategy IC of MBARPs follows from the fact that each of the above three classes are also dominant-strategy IC.

While MBARPs are a narrower class than the three others, it is general enough to incorporate the ideas of reserve pricing and bundling, which are known to increase revenue in many settings. Specifically, MBARP calls VCG on an enlarged set of agents by adding the seller—who has a reserve valuation for each bundle—and a fake agent who receives nothing and has no valuation for any item or bundle, but has a valuation for pure bundling allocations (i.e., ones where both items are allocated to the same (any) bidder).

As we show, one striking feature of MBARP is that its revenue can be represented in a simple closed form as a function of its parameters. Thus, we can solve first-order conditions on the parameters and obtain the optimal MBARP. We give a system of equations of the optimal parameters in general and solve it for some simple yet common settings: for instance, in the most trivial setting, where there are two agents and each agent's valuation for each item is drawn uniformly on [0,1], our optimal auction yields expected revenue 0.871, which significantly outperforms VCG (0.667), separate Myerson auctions (0.833) (i.e, selling the two items via a sequence of separate optimal single-item auctions), pure-bundling Myerson auction (0.800) and mixed bundling auction without reserve prices (0.786). More surprisingly, it even outperforms the optimal empirical results returned by sampling and approximation on its supersets AMA (0.860) and VVCA (0.838) [15].

We wish to emphasize the generality of our approach to analyz-

[*]This material is based upon work supported by the National Science Foundation under grants IIS-0905390, IIS-0964579, and CCF-1101668.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

ing the problem. We start from one bidder's (say i's) perspective, find critical values of the other bidder's (say j's) valuations such that these critical values completely partition j's type spaces while within each partition, the expected revenue of i can be represented with the same function of the auction parameters. The expected revenue from i is then the sum/integration of revenues from all parts of that partition. Another aspect of the generality of our approach is that we reduce the revenue maximization problem of an N-agent auction to that of a 2-agent auction. This can be achieved by, again from one bidder's perspective, using a fake agent to simulate the maximal values of other bidders' valuations. This extends and generalizes Riley and Samuelson's analysis of optimal 1-item auctions [13] to CAs.

2. RELATED RESEARCH

This section reviews in more detail well-known classes of auctions related to this paper.

2.1 The VCG mechanism and reserve prices

The most famous mechanism in CAs is the *Vickrey-Clarke-Groves mechanism (VCG)* [17, 2, 4], where the welfare maximizing allocation is chosen and each bidder i pays the sum of the others' valuations had i not participated minus the sum of the others' actual valuations. It is not hard to see that the VCG can yield revenue arbitrarily far from optimal:

EXAMPLE 2.1. *This can be seen even in one-item one-bidder setting where the bidder's valuation for the item is uniform on [0,1]. The second-price auction (aka Vickrey auction) would give the item to the bidder and obtain 0 revenue since there is no competition. The optimal auction (aka Myerson auction) would offer the item at price of 0.5 and obtain expected revenue of* $\frac{1}{2} \times 0.5 = 0.25$.

The VCG can be complemented with *reserve prices* to increase revenue. There are several definitions of reserve prices in CAs. We consider one that is most commonly seen: the seller pretends to have some reserve valuations for the different bundles, and the VCG with reserve prices is simply to apply the VCG on the set of all agents including the seller.

2.2 λ-auction

Another well-known technique to increase revenue in CAs is via bundling of items [12]. One notable example of bundling in CAs is the λ-auction [6]. It is the VCG, but with a fake bidder who does not receive any items, but has valuations towards allocations (instead of bundles).

In this paper, we consider one special subset of λ-auctions where the fake bidder is only interested in the allocations where the entire set of items is allocated to (any) one bidder. The resulting auction works as if it gave a discount for the bidders who are interested in the whole bundle. This auction is called a *mixed-bundling auction (MBA)*. Mixed-bundling auctions can also be complemented with reserve prices, which are defined by further including the seller into consideration. This class is called *mixed-bundling auction with reserve prices (MBARP)*. We will give a formal definition later in Section 4.

Jehiel et. al. used certain local arguments to show that some mixed bundling auction must yield higher revenue than any pure-bundling auction (where the only valid allocations are those that give the whole bundle to one agent) and any welfare-maximization auction [6]. We operate in a different direction, by directly calculating the closed-form expression of the expected revenue and obtaining the optimal revenue within MBARPs. As we shall see, our analysis and evaluations confirm the arguments of Jehiel et. al.

2.3 Myerson's auction, affine maximizer auctions (AMAs), and virtual valuations combinatorial auctions (VVCAs)

The third intuition to improve revenue in CAs is via virtual valuation [11]. The idea is to transform bidder's valuation by some function and then cast the welfare maximization on the virtual valuations; each bidder pays the lowest valuations he could have reported and still won his current bundle. In principle, the function can be anything, as long as it preserves the IC constraint. For example, for Bayes-Nash IC, Myerson's virtual transformation is defined as $\tilde{v}_i = v - \frac{1-F_i(v_i)}{F_i'(v_i)}$, where v_i is agent i's valuation and F_i is the cumulative distribution function of v_i.

However, for dominant-strategy IC, it turns out that affine transformations of the valuation are the only ones that satisfy the IC constraint for an unrestricted valuation space [14]. Affine maximizer auctions (AMAs) apply some affine transformation to get the virtual valuations, and then use the VCG on those virtual valuations (a fake bidder with valuations for the allocations is also included in the bidder set). Formally, the allocation rule in an AMA is to maximize $\sum_{i=0}^{n} \mu_i v_i(a) + \lambda(a)$, where μ's are constants and $\lambda(a)$ are the fake bidder's valuation for allocation a. Clearly, λ-auctions are AMAs where the affine transformation is the identity function $f(v) = v$, i.e., $\mu_i = 1$ for all i.

Virtual valuations combinatorial auctions (VVCAs) [8, 9, 15] are AMAs with the restriction $\lambda(a) = \sum_i \lambda_i(a_i)$, where $\lambda_i(a_i)$ only depends on what bidder i receives.

In summary, MBARPs are a subset of VVCAs and of λ-auctions, which, in turn, are subsets of AMAs.

2.4 Approximation results

An optimal auction may involve reserve pricing, bundling, and virtual valuations. How much revenue can one obtain with auctions that are simpler in form? This question motivated a recent line of research that focuses on designing simple auctions that yield revenue within a factor of optimal. Likhodedov and Sandholm [9, 15] give a logarithmic approximation of optimal multi-item auctions with a variation of the VCG, for two classes of settings: (1) additive valuations (where each bidder's valuation for a bundle is the sum of his valuations for the items in the bundle), and (2) unlimited supply (such as in digital music stores).

Recall that, in symmetric settings (settings where valuation distributions are identical across bidders), Myerson's auction coincides with a Vickrey auction [17] with the so-called monopoly reserve (i.e., a reserve valuation at which Myerson's virtual valuation function equals 0). Hartline and Roughgarden show that in the asymmetric single-parameter environment, the optimal auction, which is Myerson's auction, can be 2-approximated by a Vickrey auction with monopoly reserve prices [5]. Tang and Sandholm [16] study Levin's setting for complements [7] and prove that optimal revenue can be 2-approximated by using monopoly reserve price to curtail the allocation set, followed by welfare-maximizing allocation and Levin's payment rule. They also show that the optimal revenue can be 6-approximated even if the "reserve pricing" is required to be symmetric across bidders. Chawla et. al. prove several constant bounds of approximating optimal auctions in multi-dimensional type space (even though the optimal auction is unknown) using sequential posted prices [1].

3. THE SETTING

We consider a setting with one seller who has two indivisible distinguishable items for sale. There are a set $N = \{1, 2, \ldots, n\}$ of bidders. Each bidder $i \in N$ has a valuation v_1^i for the first item,

v_2^i for the second item, and valuation $v_1^i + v_2^i$ for the bundle of both items. The seller has zero valuation for any bundle. (However, she can pretend she has reserve valuations to obtain a higher revenue.)

An allocation is denoted by a vector $\vec{x}^i = (x_1^i, x_2^i)$ for each bidder i, where $x_j^i \in \{0,1\}$ is the amount of item j allocated to i. The payment from bidder i to the seller is denoted by p_i. Given a $\vec{x}^i$ and p_i, bidder i's utility function is

$$u_i(\vec{x}^i, p_i) = v_1^i x_1^i + v_2^i x_2^i - p_i$$

This means that the bidders have quasi-linear, additive utility functions and no externalities.

As usual, each v_b^i is treated by all but i as a random variable distributed on $[0,1]$ according to a cumulative distribution function F_b^i, which admits a non-negative and bounded density function f_b^i. We assume that $F_b^i = F_b^j$ for all $i, j \in N$ and all the $v's$ are independent. Our approach also applies to settings where v_1^i and v_2^i are dependent and $F_b^i \neq F_b^j$. That is, v_1^i and v_2^i are distributed according to some joint cumulative distribution $F^i(v_1^i, v_2^i)$. We make these assumptions only for the ease of presentation. We also use the standard information model where each bidder i knows his own type $v^i = (v_1^i, v_2^i)$ but others do not. The distribution over types is common knowledge.

By the revelation principle, it suffices to consider the set of direct-revelation auctions, which begin by soliciting a type from each bidder and then specify an allocation and payment for each bidder. We shall consider the direct-revelation auctions that are dominant-strategy IC and ex-post IR. A mechanism is *(weakly) dominant-strategy IC* if misreporting one's type cannot yield a higher utility for the bidder, no matter what other bidders report. A mechanism is *ex-post IR* if participation yields a non-negative utility, no matter what other bidders report.

4. MIXED-BUNDLING AUCTION WITH RESERVE PRICES

In this section, we describe the *mixed-bundling auction with reserve prices (MBARP)*. As mentioned, it is a subclass of AMA, λ-auction and VVCA. One remarkable feature of MBARP is that its expected revenue can be written in closed form as a function of its parameters, and we can optimize its revenue over the parameters.

Treat the seller as a special bidder, indexed 0, with reserve value a for item 1, b for item 2, and $a+b$ for both. Define another fake bidder, indexed $n+1$, who is never allocated any item, but has value c if some bidder $i \in N$ is allocated both items. MBARP is VCG executed on this extended set of agents $N = \{0, 1, \ldots, n+1\}$.

DEFINITION 4.1. *Given a type profile $\vec{v} = (\vec{v}^1, \ldots, \vec{v}^n)$, MBARP is defined by its allocation rule $\vec{x} = (\vec{x}^1, \ldots, \vec{x}^n)$ and payment rule $\vec{p} = (p^1, \ldots, p^n)$.*

- *The allocation rule $\vec{x}$ is,*

$$\vec{x}(\vec{v}) = \text{arg max}_{\vec{x}} = \{\sum_{j=0}^{n}(v_1^j x_1^j + v_2^j x_2^j) + v^{n+1}(\vec{x})\}.$$

By definition, $v^{n+1}(\vec{x}) = c$ if $x_1^i = x_2^i = 1$ for some $1 \leq i \leq n$ and $v^{n+1}(\vec{x}) = 0$ otherwise.

- *The payment rule $\vec{p}$ is,*

$$p^i(\vec{v}) = -(\sum_{j\neq i}^{n}(v_1^j x_1^j + v_2^j x_2^j) + v^{n+1}(\vec{x})) + \sum_{j\neq i}^{n}(v_1^j \tilde{x}_1^j + v_2^j \tilde{x}_2^j) + v^{n+1}(\vec{\tilde{x}}),$$

where $\vec{\tilde{x}}(\vec{v}) = \vec{x}^{-i}(\vec{v}^{-i})$. In other words, $\vec{\tilde{x}}$ is the welfare-maximizing allocation without the participation of i.

LEMMA 4.1.

1. *MBARP is dominant-strategy IC and* ex post *IR.*
2. *A bidder with the lowest type (that is, $v_1^i = v_2^i = 0$) gets 0 utility.*
3. *Given the allocation rule of MBARP, this particular payment rule, $\vec{p}$, yields the highest revenue among all the payment rules that satisfy 1 and 2.*

Proof: The proof of 1 and 2 mirror those of the VCG. The proof of 3 follows from [6, Lemma 1]. ■

5. OPTIMAL PARAMETERS FOR THIS AUCTION CLASS

In this section, we develop a systematic approach to calculating the revenue of MBARP. This approach is general and can be applied to other strategy-proof two-item auctions.

In what follows, we first analyze the case with two bidders and then reduce those with more than two bidders to the two-bidder case.

5.1 Two-bidder setting

Suppose there are two bidder, i and j. Given the values of a, b, and c, as well as j's report (v_1^j, v_2^j), we compute the formula for i's expected payment. For this purpose, let us first look at i's allocation space. The definition of MBARP (the allocation rule and the payment rule) implies that i's allocation space must be of the shape in Figure 1. Formally:

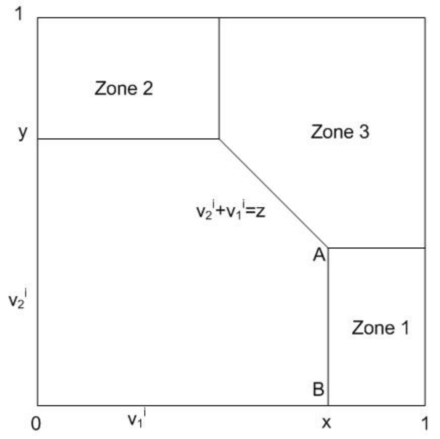


Figure 1: Bidder i's allocation space.

LEMMA 5.1.

- *The allocation space of bidder i is of exactly the shape as in Figure 1, i.e., lines are either vertical, horizontal or of -1 slope, with x, y, z being undecided variables.*
- *Bidder i gets item 1 for the price of x when his type is in Zone 1, gets item 2 for the price of y when his type is in Zone 2 and gets both items for the price of z when his type is in Zone 3.*

- *The expected payment of i is*

$$R(x,y,z) = x \cdot \int_{(v_1^i,v_2^i)\in 1} f_1^i(v_1^i) f_2^i(v_2^i) dv_1^i dv_2^i + y \cdot \int_{(v_1^i,v_2^i)\in 2} f_1^i(v_1^i) f_2^i(v_2^i) dv_1^i dv_2^i + z \cdot \int_{(v_1^i,v_2^i)\in 3} f_1^i(v_1^i) f_2^i(v_2^i) dv_1^i dv_2^i.$$

Proof: The first two claims follow from an exhaustive case analysis immediately following this lemma. Given the first two claims, the third one is straightforward. ■

5.1.1 *First part of the analysis:* $a \geq c$ *and* $b \geq c$

In this section we consider those parameter values that satisfy $a \geq c$ and $b \geq c$. We will consider the other parameter values in the next section.

We now determine the values of x, y, and z case by case.

Case 1. $v_1^j \geq a, v_2^j \geq b$. By the allocation rule of MBARP, bidder i gets the bundle iff

$$v_1^i + v_2^i + c \geq v_1^j + v_2^j + c \text{ and}$$
$$v_1^i + v_2^j \leq v_1^i + v_2^i + c \text{ and}$$
$$v_2^i + v_1^j \leq v_1^i + v_2^i + c.$$

The first equation ensures that bidder j does not get the bundle and the second (third) is to ensure that bidder i does not end up with item $1(2)$ only. These constraints give Zone 3 in Figure 2. By the definition of MBARP, bidder i pays $z = v_1^j + v_2^j$ in this region.

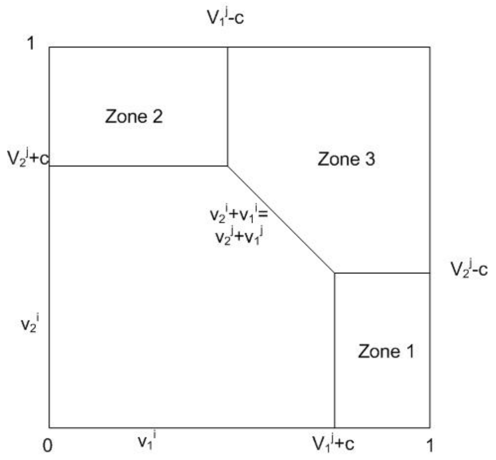


Figure 2: Case 1.

Similarly, bidder i gets item 1 only iff

$$v_1^i + v_2^j \geq v_1^j + v_2^j + c \text{ and}$$
$$v_1^i + v_2^j > v_1^i + v_2^i + c.$$

The first equation ensures that bidder j does not get the bundle and the second ensures that i does not get the bundle. These constraints give Zone 1 in Figure 2. Bidder i pays $x = v_1^j + c$.

Similarly, we can derive Zone 2 and $y = v_2^j + c$.

To sum up, the allocation space of bidder i in this case is given by Figure 2. We have $x = v_1^j + c$, $y = v_2^j + c$ and $z = v_1^j + v_2^j$.

Case 2. $v_1^j < a, v_2^j < b$.

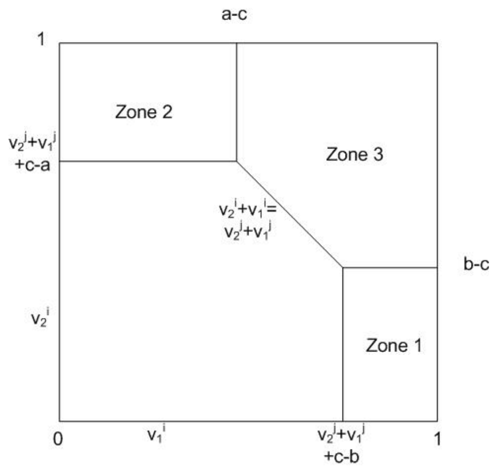


Figure 3: Case 2a.

- **Subcase a**. $v_1^j + v_2^j + c \geq a + b$. By the allocation rule of MBARP, bidder i gets the bundle iff

$$v_1^i + v_2^i + c \geq v_1^j + v_2^j + c \text{ and}$$
$$v_1^i + b \leq v_1^i + v_2^i + c \text{ and}$$
$$v_2^i + a \leq v_1^i + v_2^i + c.$$

  Thus, $z = v_1^j + v_2^j$.

  Bidder i gets item 1 only iff

$$v_1^i + b \geq v_1^j + v_2^j + c \text{ and}$$
$$v_1^i + b > v_1^i + v_2^i + c.$$

  Thus, $x = v_1^j + v_2^j + c - b$. Symmetrically, we have $y = v_1^j + v_2^j + c - a$.

  To sum up, the allocation space is given in Figure 3. We have $x = v_1^j + v_2^j + c - b$, $y = v_1^j + v_2^j + c - a$ and $z = v_1^j + v_2^j$.

- **Subcase b**. $v_1^j + v_2^j + c < a + b$. Bidder i gets the bundle iff

$$v_1^i + v_2^i + c \geq a + b \text{ and}$$
$$v_1^i + b \leq v_1^i + v_2^i + c \text{ and}$$
$$v_2^i + a \leq v_1^i + v_2^i + c.$$

  Thus, $z = a + b - c$.

  Bidder i gets item 1 only iff

$$v_1^i + b \geq a + b \text{ and}$$
$$v_1^i + b > v_1^i + v_2^i + c.$$

  Thus, $x = a$. Symmetrically, $y = b$. The allocation space is given in Figure 4.

  We have $x = a$, $y = b$ and $z = a + b - c$.

Case 3. $v_1^j < a, v_2^j \geq b$.

- **Subcase a**. $v_1^j + v_2^j + c \geq a + v_2^j$. Bidder i gets the bundle iff

$$v_1^i + v_2^i + c \geq v_1^j + v_2^j + c \text{ and}$$
$$v_1^i + v_2^j \leq v_1^i + v_2^i + c \text{ and}$$
$$v_2^i + a \leq v_1^i + v_2^i + c.$$

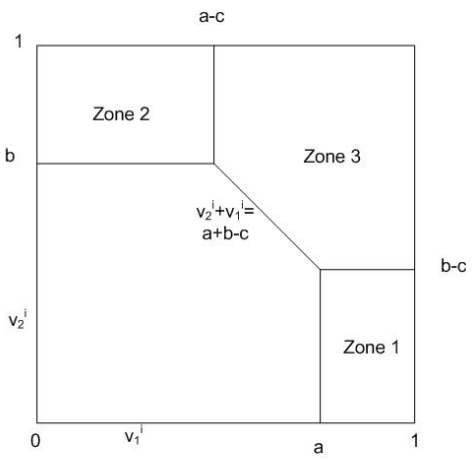


Figure 4: Case 2b.

Thus, $z = v_1^j + v_2^j$.

Similarly, $x = v_1^j + c$ and $y = v_1^j + v_2^j + c - a$.

The allocation space is given in Figure 5.

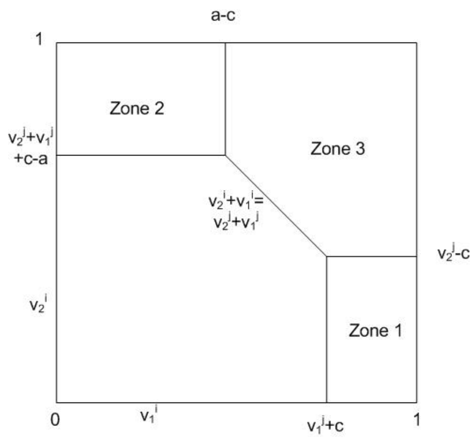


Figure 5: Case 3a.

We have $x = v_1^j + c$, $y = v_1^j + v_2^j + c - a$ and $z = v_1^j + v_2^j$.

- **Subcase b**. $v_1^j + v_2^j + c < a + v_2^j$. Bidder i gets the bundle iff

$$v_1^i + v_2^i + c \geq a + v_2^j \text{ and}$$
$$v_1^i + v_2^i \leq v_1^i + v_2^i + c \text{ and}$$
$$v_2^i + a \leq v_1^i + v_2^i + c.$$

Thus, $z = a + v_2^j - c$.

Similarly, $x = a$ and $y = v_2^j$.

The allocation space is given in Figure 6.

We have $x = a$, $y = v_2^j$ and $z = a + v_2^j - c$.

Case 4. $v_1^j \geq a, v_2^j < b$. This case is symmetric to Case 3.

- **Subcase a**. $v_1^j + v_2^j + c \geq b + v_1^j$. We have $y = v_2^j + c$, $x = v_1^j + v_2^j + c - b$ and $z = v_1^j + v_2^j$.

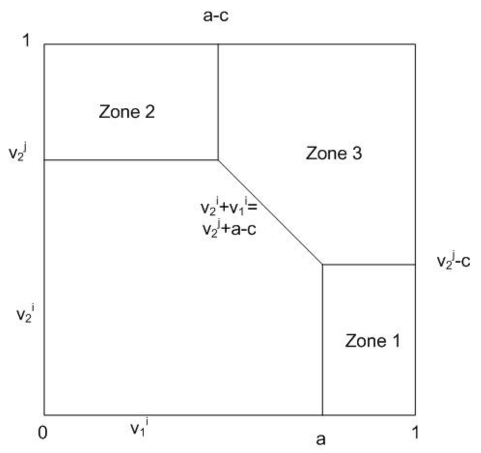


Figure 6: Case 3b.

- **Subcase b**. $v_1^j + v_2^j + c < b + v_1^j$. We have $y = b$, $x = v_1^j$ and $z = b + v_1^j - c$.

The above case-by-case analysis forms a complete partition of bidder j's type space, as shown in Figure 7, with Case 1 corresponding to Zone P_1, Case 2, Subcase a corresponding to Zone P_{2a}, and so on.

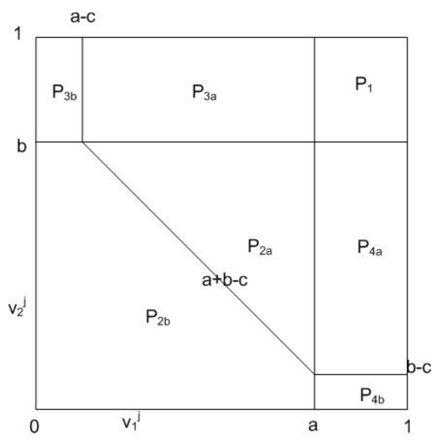


Figure 7: Partition of bidder j's type space.

We are now ready to give a closed form expression for bidder i's expected payment $r_0^i(a, b, c)$.

$$r_0^i(a,b,c) =$$
$$\int_{(v_1^j,v_2^j)\in P_1} f_1^j(v_1^j)f_2^j(v_2^j)R(v_1^j+c, v_2^j+c, v_1^j+v_2^j)dv_1^j dv_2^j$$
$$+ \int_{(v_1^j,v_2^j)\in P_{2a}} f_1^j(v_1^j)f_2^j(v_2^j)R(v_1^j+v_2^j+c-b,$$
$$v_1^j+v_2^j+c-a, v_1^j+v_2^j)dv_1^j dv_2^j$$
$$+ \dots$$
$$+ \int_{(v_1^j,v_2^j)\in P_{4b}} f_1^j(v_1^j)f_2^j(v_2^j)R(v_1^j, b, b+v_1^j-c)dv_1^j dv_2^j.$$

In the symmetric setting (where $f^i = f^j$), bidder j's expected payment also equals $r_0^i(a, b, c)$. This leads to the following theorem.

THEOREM 1. *For $a \geq c, b \geq c$,*

- *The expected revenue of MBARB is $2r_0^i(a, b, c)$;*
- *The optimal parameters a, b, c are given by*[1]

$$\frac{\partial r_0^i(a,b,c)}{\partial a} = \frac{\partial r_0^i(a,b,c)}{\partial b} = \frac{\partial r_0^i(a,b,c)}{\partial c} = 0.$$

In the asymmetric 2-bidder setting, $r_0^j(a, b, c)$ can be obtained by swapping the role of i and j in $r_0^i(a, b, c)$. Then the expected revenue of MBARP is $r_0^i(a, b, c) + r_0^j(a, b, c)$.

5.1.2 The remaining parts of the analysis: $(a < c,\ b \geq c)$ or $(a \geq c,\ b < c)$ or $(a < c,\ b < c)$

Now let us reexamine Figure 7. The existence of P_{3b} relies on the fact that $a - c \geq 0$. The underlying explanation is that when $a < c$, the condition of Case 3, Subcase b, which is $v_1^j + v_2^j + c < a + v_2^j$, never holds. In other words, the condition of Case 3, Subcase a holds trivially. This reasoning leads to the reduction of Figure 7 into Figure 8 when $a < c$ and $b \geq c$.

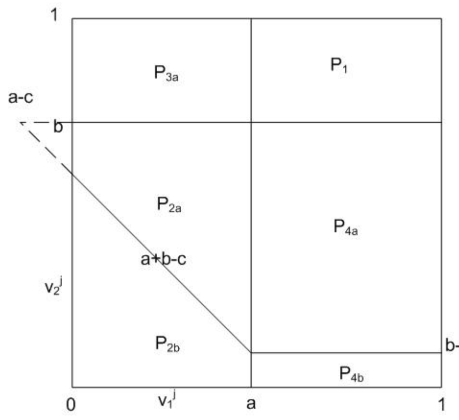


Figure 8: Partition of bidder j's type space when $a < c$ and $b \geq c$.

The corresponding x's y's and z's in each case are still the same as those when $a \geq c$ and $b \geq c$, except that Case 3b does not exist. We denote the expected revenue from i in this case by $r_1^i(a, b, c)$.

For the same reason, we can get revenue $r_2^i(a, b, c)$ when $a \geq c$ and $b < c$.

When $a < c$ and $b < c$, we need to further distinguish two subcases: $a+b-c \geq 0$ and $a+b-c < 0$, since when $a+b-c < 0$, P_{2b} also diminishes.

We denote the revenues for the two subcases above by $r_3^i(a, b, c)$ and $r_4^i(a, b, c)$.

So, the maximal revenue is given by $max_{k=0,\dots,4}\{r_k^i(a, b, c)\}$. This completes the analysis of the two-bidder case.

5.2 More than two bidders

The allocation space grows rapidly as the number of bidders increases. The case-by-case enumeration becomes unmanageable even with 3 bidders. However, by applying order statistics, we are able to calculate bidder i's expected revenue, by treating all the other bidders simply as one bidder whose valuation for items is distributed according to the $(N-1)$th-order statistic of the valuations from these bidders. In other words, what really matters to bidder i is the greatest valuation of the remaining bidders on each bundle.

In what follows, we shall consider the symmetric setting only. That is, bidders' values are drawn from the same distributions: $F_1(v_1)$ for the first item and $F_2(v_2)$ for the second item. We denote by $F_1^k(v_1^k)$ the (k)th (with $(N-1)$th being the greatest) order statistic of $N-1$ i.i.d variables drawn from $F_1(v_1)$. Analogously, we denote by $F_2^k(v_2^k)$ the (k)th order statistic of $N-1$ i.i.d variables drawn from $F_2(v_2)$. We denote by $F_{sum}^k(v_{sum}^k)$ the (k)th order statistic of $N-1$ i.i.d variables which are sums of $N-1$ pairs of variables, one drawn from $F_1(v_1)$ and the other drawn from $F_2(v_2)$.

Fixing realized valuations v_1^{N-1} and v_1^{N-1}, and treating them as valuations of a fake bidder j for the two items respectively, we have two cases to consider:

- **Case 1.** v_1^{N-1} and v_2^{N-1} are realized by the same actual bidder. This happens with probability $p_1 = \frac{1}{N-1}$ since all the bidders are symmetric. In this case, we can completely reduce the analysis to our previous 2-bidder case, with $v_1^j = v_1^{N-1}$ and $v_2^j = v_2^{N-1}$. We denote the expected revenue from bidder i for this realization by $R_1^i(a, b, c)|(v_1^{N-1}, v_2^{N-1})$.
- **Case 2.** v_1^{N-1} and v_2^{N-1} are realized by different actual bidders. This happens with probability $p_2 = \frac{N-2}{N-1}$. There are two subcases.
  - **Subcase 1.** $v_{sum}^{N-1} + c \leq v_1^{N-1} + v_2^{N-1}$. This happens with probability $p_{21} = F_{sum}^{N-1}(v_1^{N-1} + v_2^{N-1} - c)$. We can reduce the analysis to the 2-bidder case, with a slight modification where the bundling parameter c for bidder j is 0 if he gets both items and bidder i remains unchanged. We can still go though a similar case-by-case analysis of bidder i's expected revenue since Lemma 5.1 still holds. (For example, (x, y, z) in Case 1 should now be changed to $(v_1^{N-1}, v_2^{N-1}, v_1^{N-1} + v_1^{N-1} - c)$. Similarly, one can derive x's, y's, and z's in other cases.) We denote the expected revenue from bidder i for this part by $R_{21}^i(a, b, c)|(v_1^{N-1}, v_2^{N-1})$. This is an expectation over v_{sum}^{N-1} as well as over i's own valuations.
  - **Subcase 2.** $v_{sum}^{N-1} + c > v_1^{N-1} + v_2^{N-1}$ but $v_{sum}^{N-1} < v_1^{N-1} + v_2^{N-1}$. This happens with probability $p_{22} = F_{sum}^{N-1}(v_1^{N-1} + v_2^{N-1}) - F_{sum}^{N-1}(v_1^{N-1} + v_2^{N-1} - c)$. This case means that to get the bundle, i must have a valuation no less than $v_{sum}^{N-1} + c$ (instead of $v_1^{N-1} + v_2^{N-1}$). This reduces the analysis to a 2-bidder auction where the seller's valuations are given by $(a, b, \max\{a + b, v_{sum}^{N-1} + c\})$. The analysis is also similar to that in Section 5. Denote the expected revenue from bidder i for this part by $R_{22}^i(a, b, c)|(v_1^{N-1}, v_2^{N-1})$.

Therefore, the expected revenue from bidder i is

$$\begin{aligned}
&\int_{(v_1^{N-1}, v_2^{N-1})} f_1^{N-1}(v_1^{N-1}) f_2^{N-1}(v_2^{N-1}) \\
&(p_1 R_1^i(a, b, c)|(v_1^{N-1}, v_2^{N-1}) + \\
&p_2 R_{21}^i(a, b, c)|(v_1^{N-1}, v_2^{N-1}) + \\
&p_2 R_{22}^i(a, b, c)|(v_1^{N-1}, v_2^{N-1}))dv_1^{N-1} dv_2^{N-1}.
\end{aligned}$$

[1]In general, the revenue expression is not necessarily convex and the equations have multiple roots. Therefore, it is necessary to evaluate at each root as well as boundaries to determine the optimal set of parameters. These can be easily done in Mathematica 7, as shown in our simulation in Section 6.

To evaluate the integration above (like in the two-bidder case), we partition j's valuation space and integrate each part separately, since $R_1^i(a,b,c)$, $R_{21}^i(a,b,c)$ and $R_{22}^i(a,b,c)$ take different form for different (v_1^{N-1}, v_2^{N-1}). However, each case above (Case 1, Case 2a, and Case 2b) will give us a different partition. Thus, we actually partition j's valuation space into the *coarsest common refinement* of all the partitions generated by the cases above.

The optimal auction parameters a, b, and c then follow from the same first-order conditions as given in Theorem 1.

6. COMPARISON OF AUCTIONS

This section compares the revenues of various auction classes in three different settings, all of which have two bidders and symmetric valuations:

Setting 1: $f_1(v_1) = f_2(v_2) = 1$ on [0,1].

Setting 2: $f_1(v_1) = 2v_1, f_2(v_2) = 1$ on [0,1].

Setting 3: $f_1(v_1) = 2 - 2v_1, f_2(v_2) = 1$ on [0,1].

While our analysis carries over for all (a, b, c), we restricted ourselves to $0 \leq a \leq 1$ and $0 \leq b \leq 1$ in this section. This is without loss of generality as shown in Section 7.

Auction	Optimal a, b, c	Revenue
r_0	0.577, 0.577, 0.265	0.871
r_1	0.561, 0.791, 0.561	0.848
r_2	0.791, 0.561, 0.561	0.848
r_3	0.770, 0.770, 0.770	0.843
r_4	0.257, 0.257, 1.063	0.833
VCG	0.000, 0.000, 0.000	0.667
Separate Myerson auctions	0.500, 0.500, 0.000	0.833
Pure-bundling Myerson	$r_{Bundle} = 0.816$	0.839
Mixed-bundling auction	0.000, 0.000, 0.333	0.786
AMA*	N/A	0.860
VVCA*	N/A	0.838

Table 1: Expected revenues in Setting 1.

For Setting 1, the expected revenues are listed in Table 1. r_0 to r_4 have the same meaning as in Section 5: they are the revenues of the different cases of MBARP, and the first case (r_o, which corresponds to $a \geq c, b \geq c$) yields the highest revenue in this setting. The VCG mechanism is an instance of MBARP with $a = b = c = 0$. The 8th row, separate Myerson auctions, denotes the auction where two items are sold sequentially through optimal single-item auctions (Myerson auctions). The 9th row, pure-bundling Myerson auction, denotes the optimal auction that sells both items to one agent. The 10th and 11th row, AMA* and VVCA*, denote approximated results returned by experiments [15], because there is no known analytical way to optimize the parameters of AMA and VVCA. All other revenue numbers in this section are calculated analytically (by Mathematica 7.0) and are thus (rounded) exact solutions.

We have the following conclusions:

- MBARP yields the highest revenue, even compared to experimentally “optimized" parameter settings for its supersets: AMA and VVCA.
- The reserve prices a and b play more important roles than the bundling parameter c: separate Myerson auctions yield revenue 0.833 while the mixed-bundling auction without reserve prices yields only 0.786.
- Separate Myerson auctions are inferior to the pure-bundling Myerson auction in this setting. (In fact, this is the case for all the three settings.) This is analogous to a milestone result by Palfrey [12] from the bundling literature stating that the pure-bundling Vickrey auction yields more revenue than separate Vickrey auctions in the two-bidder, N-object setting. This result also suggests an interesting general question: Does Palfrey's theorem hold for Myerson auctions as well?

For Settings 2 and 3, the expected revenues are listed in Tables 2 and 3, respectively. Again, MBARP outperforms the other auctions. Optimal expected revenues for VVCA and AMA are unknown.

Auction	a,b,c	Revenue
MBARP(r_0)	0.641, 0.581, 0.225	1.037
VCG	0.000, 0.000, 0.000	0.867
Separate Myerson auctions	0.577, 0.500, 0.000	1.001
Pure-bundling Myerson	$r_{Bundle} = 0.908$	1.002

Table 2: Expected revenues in Setting 2.

Auction	a,b,c	Revenue
MBARP(r_0)	0.415, 0.575, 0.266	0.709
VCG	0.000, 0.000, 0.000	0.533
Separate Myerson auctions	0.333, 0.500, 0.000	0.672
Pure-bundling Myerson	$r_{Bundle} = 0.694$	0.688

Table 3: Expected revenues in Setting 3.

7. DISCUSSION

In this section we discuss three additional issues.

7.1 Better starting point for automated mechanism design

Sandholm et al. show that, by starting from the VCG and then using hill-climbing algorithms to search through the parameter spaces of AMA or VVCA, the auction ends up with high-revenue (locally optimal) parameters [15, 8, 9]. Using that approach, as shown in Section 6, they obtained expected revenue 0.860 for AMA and 0.838 for VVCA in Setting 1. An easy way to improve their results is to start from the optimal MBARP instead of VCG, since optimal MBARP is also an instance of VVCA and AMA, and yields higher revenue than VCG. In fact, we implement one of the algorithms named BLAMA from [15] and improve the optimal revenue in Setting 1 to 0.872.

7.2 What if $a < 0$ or $a > 1$?

Although our theoretical analysis applies for all tuples of (a, b, c), our Mathematica program (Section 6) was based on the assumption that a and b are both within [0,1]. We now show that our assumption is without loss of generality. First, consider $a < 0$. This case is equivalent to the one where $a = 0$. (Going through the case-by-case analysis, we have only Case 1 and Case 4 left, and both of those cases yield x's, y's and z's that do not depend on a.) Intuitively, this means there is no reserve price on item 1.

Now consider $a > 1$. This case is equivalent to the one where we have revised auction parameters a' and c' as follows: $a' = 1$ and $c' = c - (a - 1)$. (Going through the case-by-case analysis, we have only Case 2 and Case 3, and both cases yield x's, y's and z's that either do not depend on a, or depend only on $c - a$, or have $x > 1$. Each of these three cases will remain the same after we switch to a' and c'.) This means that no bidder can win exactly one item.

7.3 Simple versus optimal mechanisms

A direction that might further improve revenue is to introduce asymmetry into the allocation rule, as Myerson did in the optimal one-item auction [11]. However, there are two concerns. First, we depart from Myerson's framework in the sense that we stick to the paradigm of dominant-strategy IC, while Myerson relaxed it to Bayes-Nash IC.

The second concern is related to the simplicity of the auction. This is part of the reason that Riley and Samuelson [13] restrict themselves on symmetric one-item auctions. (Fortunately, they ended up with Myerson's auction in symmetric settings. By analogy, this suggests that the optimal two-item additive-valuations auctions might lie within the MBARP family.) This is also the major motivation why revenue lower bounds from VCG-like mechanisms have been studied [5, 16].

Because we have a closed-form expression for revenue, we can explicitly trade off revenue for simplicity. For instance, for Setting 1 we may like the simplicity of the MBARP with $(a, b, c) = (0.6, 0.6, 0.3)$. We can calculate that the revenue for this configuration is 0.8696, which is very close to the optimal MBARP. That set of parameters are also noted by Jehiel et. al. [6]. Another nice set of parameters is $(a, b, c) = (\frac{1}{2}, \frac{1}{2}, \frac{1}{4})$, and it yields revenue 0.8609.

8. CONCLUSIONS AND FUTURE WORK

Revenue-maximizing multi-item auctions are perhaps the most important open topic in auction design and mechanism design at large. It is open even in the 2-item additive case. We studied a class of two-item auctions called mixed-bundling auctions with reserve prices (MBARP). The idea is that the allocation rule is biased towards the bids for the whole bundle to increase the probability of selling the bundle together. It also includes reserve pricing to further increase revenue. In fact, it is general enough to include auctions such as the VCG and separate Myerson auctions. A remarkable feature of MBARPs is that the expected revenue can be represented in a simple closed-form expression, and can be optimized easily. We gave a system of equations of the optimal parameters in general and solved it for some canonical settings. The optimal MBARP yields significantly higher revenue than the known auction classes with closed-form revenue expressions. Furthermore, its revenue even exceeds the optimal empirical results returned by sampling and approximation on much broader classes, where the truly optimal expected revenues are difficult to obtain.

There are several directions for future research. We considered the case where the bidders' valuations are additive. It is not hard to see that, with appropriate rotations of the lines in Lemma 5.1, our current approach carries over to calculating the expected revenue when bidders' valuations for the bundle are linear combination of their valuations for the items in that bundle (formally, $\lambda_1 v_1^i + \lambda_2 v_1^i + \lambda_3$, where $\lambda_1, \lambda_2, \lambda_3$ are constants). Perhaps our approach can be generalized even further.

Second, can the bundling parameter, c, be optimized separate from the reserve prices a and b?

Third, can we extend this approach to more than two items? For example, when there are 3 items, we need at least 4 bundling parameters: 3 for each pair of items and 1 for the whole bundle. Does there exist a simple derivation from the optimal 2-item parameters to the optimal 3-item parameter? More generally, can we reduce the analysis of three items to that of two items?

Future work also includes introducing asymmetry into the allocations rule. This might increase revenue even when bidders' valuations are symmetrical. Or, does the revenue-maximizing mechanism lie within the MBARP family?

9. REFERENCES

[1] S. Chawla, J. D. Hartline, D. L. Malec, and B. Sivan. Multi-parameter mechanism design and sequential posted pricing. STOC '10, pages 311–320. ACM, 2010.

[2] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 2:19–33, 1971.

[3] V. Conitzer and T. Sandholm. Self-interested automated mechanism design and implications for optimal combinatorial auctions. In *ACM EC*, pages 132–141, 2004.

[4] T. Groves. Incentives in Teams. *Econometrica*, 41:617–631, 1973.

[5] J. D. Hartline and T. Roughgarden. Simple versus optimal mechanisms. In *ACM EC*, 2009.

[6] P. Jehiel, M. Meyer-Ter-Vehn, and B. Moldovanu. Mixed bundling auctions. *Journal of Economic Theory*, 127(1):494–512, 2007.

[7] J. Levin. An optimal auction for complements. *Games and Economic Behavior*, 18(2):176–192, February 1997.

[8] A. Likhodedov and T. Sandholm. Methods for boosting revenue in combinatorial auctions. In *AAAI*, pages 232–237, 2004.

[9] A. Likhodedov and T. Sandholm. Approximating revenue-maximizing combinatorial auctions. In *AAAI*, pages 267–273, 2005.

[10] E. Maskin and J. Riley. Optimal multi-unit auctions. In *The Economics of Missing Markets, Information, and Games*, chapter 14, pages 312–335. 1989.

[11] R. B. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981.

[12] T. R. Palfrey. Bundling decisions by a multiproduct monopolist with incomplete information. *Econometrica*, 51(2):463–83, March 1983.

[13] J. G. Riley and W. F. Samuelson. Optimal auctions. *American Economic Review*, 71(3):381–92, June 1981.

[14] K. Roberts. The characterization of implementable social choice rules. In J.-J. Laffont, editor, *Aggregation and Revelation of Preferences*. North-Holland Publishing Company, 1979.

[15] T. Sandholm, A. Likhodedov, and A. Gilpin. Automated design of revenue-maximizing combinatorial auctions. *Operations Research*. Special issue on Computational Economics. To appear.

[16] P. Tang and T. Sandholm. Approximating optimal combinatorial auctions for complements using restricted welfare maximization. In *IJCAI*, pages 379–385, 2011.

[17] W. Vickrey. Counterspeculation, Auctions and Competitive Sealed Tenders. *Journal of Finance*, pages 8–37, 1961.

Eliciting Forecasts from Self-interested Experts: Scoring Rules for Decision Makers

Craig Boutilier
Department of Computer Science
University of Toronto, Toronto, CANADA
cebly@cs.toronto.edu

ABSTRACT

Scoring rules for eliciting expert predictions of random variables are usually developed assuming that experts derive utility only from the quality of their predictions. We study more realistic settings in which (a) the principal is a decision maker who takes a decision based on the expert's prediction; and (b) the expert has an inherent *interest* in the decision. Not surprisingly, in such situations, the expert usually has an incentive to misreport her forecast to influence the choice of the decision maker. We develop a general model for this setting and introduce the concept of a *compensation rule*. When combined with the expert's inherent utility for decisions, a compensation rule induces a *net scoring rule* that behaves like a traditional scoring rule. Assuming full knowledge of expert utility, we provide a complete characterization of all (strictly) proper compensation rules. We then analyze the case when the expert's utility function is not fully known to the decision maker. We show bounds on: (a) expert incentive to misreport; (b) the degree to which an expert will misreport; and (c) decision maker loss in utility due to such uncertainty. These bounds depend in natural ways on the degree of uncertainty, the local degree of convexity of net scoring function, and properties of the decision maker's utility function. Finally, we briefly discuss the use of compensation rules in prediction markets.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Multiagent Systems

General Terms

Algorithms, Economics, Theory

Keywords

prediction markets, mechanism design, incentives, decision theory

1. INTRODUCTION

Eliciting predictions of uncertain events from knowledgeable *experts* is a fundamental problem in statistics, economics, operations research, artificial intelligence and a variety of other areas [18, 4]. Increasingly, robust mechanisms for prediction are being developed, proposed and/or applied in real-world domains ranging from elections and sporting events, to events of public interest (e.g., disease spread or terrorist action), to corporate decision making. Indeed, the very idea of crowd-sourcing and information (or prediction) markets is predicated on the existence of practical mechanisms for information elicitation and aggregation.

Prediction mechanisms must provide an expert agent with incentives to reveal a forecast they believe to be accurate. Many forms of "outcome-based" *scoring rules*, either individual or market-based, provide experts with incentives to: (a) provide sincere forecasts; (b) invest effort to improve the accuracy of their personal forecasts; and (c) participate in the mechanism if they believe they can improve the quality of the principal's forecast. However, with just a few exceptions (see, e.g., [14, 12, 17, 15, 2, 6, 8, 5]), most work fails to account for the ultimate use to which the forecast will be put. Furthermore, even these models assume that the experts who provide their forecasts derive no utility from the final forecast, or how it will be used, except insofar as they will be rewarded by the prediction mechanism itself.

In many settings, this assumption is patently false: the principal is will often exploit the elicited forecast in order to make a *decision* [10, 12, 15, 2, 5]. In corporate prediction markets, the principal may base strategic business decisions on internal predictions of uncertain events. In a hiring committee, the estimated probability of candidates accepting offers influences the order in which (and whether) offers are made. Providing appropriate incentives in the form of scoring rules is often difficult in such cases [15, 2, 6]. However, just as critically, experts often have *their own interests* in specific decisions. For example, in corporate settings, an expert from a certain division may have an incentive to misreport demand for specific products, thus influencing R&D decisions that favor her unit. In a hiring committee, a member may misreport the odds that a candidate will accept a competing position in order to bias the "offer strategy" in a way that favors her preferred candidate.

In this work, we develop a formal model of scoring rules that incentivize truthful forecasts even when experts have an interest in the decisions taken by the principal. Naturally such rules must compensate experts for "sacrificing their own interests." One might respond by ignoring forecasts from experts with such conflicts. Unfortunately, decision makers often *must rely on the advice and predictions of experts who have some stake in their decisions*. This is especially true in organizations (e.g., corporate R&D decisions, faculty hiring, etc.), when advice from employees or group members is solicited; but it also holds in any case where the principal is *uncertain* about an expert's true interests. Our model and analysis, rather than ignoring the issue and hoping for the best, provides insight into how best to reward experts for their forecasts. Other work has studied both decision making and incentive issues in prediction markets [14, 10, 15, 2, 8, 7, 17], but only rarely addresses the natural question of expert self-interest in decisions [12, 5].

Our basic building block is a scoring rule for a single expert who knows the principal's *policy*—i.e., mapping from forecasts to decisions—and where the principal knows the expert's utility for decisions. We show that the scoring rule must compensate the expert in a simple, intuitive way based on her utility function: we use a *compensation function* that induces a proper scoring rule. We provide a complete characterization of proper compensation functions,

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

as well those which, in addition, satisfy *participation constraints*.

We then analyze *expert uncertainty* in the principal's policy, and *principal uncertainty* in the expert's utility. First, we note that the expert need not know the principal's policy prior to providing her forecast as long as she can verify which decision has been taken after the fact. Second, we observe that, in general, the principal cannot ensure truthful reporting without full knowledge of the expert's utility function. However, principals will almost always have *some* partial knowledge of expert utility. We show that *bounds* on this uncertainty give rise bounds on each of the following: (i) the expert's incentive to misreport; (ii) the deviation of the expert's misreported forecast from her true beliefs; and (iii) the loss in utility the principal will realize due to this uncertainty. The first two bounds rely on the notion of *strong convexity* of the net scoring function. The third uses natural properties of the principal's utility function. We show that these bounds can be significantly tightened using *local* strong convexity, requiring only sufficient (and differential) convexity near the decision boundaries of the principal's policy. We conclude by briefly discussing a market scoring rule (MSR) based on our one-shot compensation rule.

2. BACKGROUND: SCORING RULES

We first review scoring rules and prediction markets (see [18, 4] for more details). We assume an agent—the *principal*—must assess the distribution of a discrete random variable $\mathcal{X}$ with domain $X = \{x_1, \ldots, x_m\}$. Let $\Delta(X)$ denote the set of distributions over X, where $\mathbf{p} \in \Delta(X)$ is a nonnegative vector $\langle p_1, \ldots, p_m \rangle$ s.t. $\sum_i p_i = 1$. The principle can engage one or more experts to provide a forecast $\mathbf{p} \in \Delta(X)$. We first assume a single expert E. Consider a running example: the chief strategy officer (CSO) of a company asks a division head to estimate demand for a new product prior to committing R&D efforts to that product.

We assume E has beliefs $\mathbf{p}$ about $\mathcal{X}$. To incentivize E to report $\mathbf{p}$ faithfully (and devote reasonable effort to developing *accurate* beliefs), a variety of *scoring rules* have been proposed [16, 13, 9]. A *scoring rule* is a function $S : \Delta(X) \times X \to \mathbb{R}$ that provides a score (or payoff) $S(\mathbf{r}, x_i)$ to E if she reports forecast $\mathbf{r}$ and the realized outcome of $\mathcal{X}$ is x_i, essentially rewarding E for her predictive "performance" [13]. If E has beliefs $\mathbf{p}$ and reports $\mathbf{r}$, her expected score is $S(\mathbf{r}, \mathbf{p}) = \sum_i S(\mathbf{r}, x_i) p_i$. We say S is a *proper scoring rule* iff a truthful report is optimal for E:

$$S(\mathbf{p}, \mathbf{p}) \geq S(\mathbf{r}, \mathbf{p}), \quad \forall \mathbf{p}, \mathbf{r} \in \Delta(X) \tag{1}$$

We say S is *strictly proper* if inequality (1) is strict for $\mathbf{r} \neq \mathbf{p}$. A popular strictly proper scoring rule is the log scoring rule, where $S(\mathbf{p}, x_i) = a \log p_i + b_i$ (for arbitrary constants $a > 0$ and b_i) [13, 16]. In what follows, we restrict attention to *regular* scoring rules in which payment $S(\mathbf{r}, x_i)$ is bounded whenever $r_i > 0$.

Proper scoring rules can be fully characterized in terms of convex cost functions [13, 16]; here we review the formulation of Gneiting and Raftery [9]. Let $G : \Delta(X) \to \mathbb{R}$ be any convex function over distributions—we refer to G as a *cost function*. We denote by $G^* : \Delta(X) \to \mathbb{R}^m$ some *subgradient* of G, satisfying

$$G(\mathbf{q}) \geq G(\mathbf{p}) + G^*(\mathbf{p}) \cdot (\mathbf{q} - \mathbf{p})$$

for all $\mathbf{p}, \mathbf{q} \in \Delta(X)$.[1] Such cost functions and associated subgradients can be used to derive any proper scoring rule.

THEOREM 1. *[13, 16, 9] A regular scoring rule S is proper iff*

$$S(\mathbf{p}, x_i) = G(\mathbf{p}) - G^*(\mathbf{p}) \cdot (\mathbf{p}) + G_i^*(\mathbf{p}) \tag{2}$$

[1]If G is differentiable at $\mathbf{p}$ then the subgradient at that point is unique, namely, the gradient $\nabla G(\mathbf{p})$.

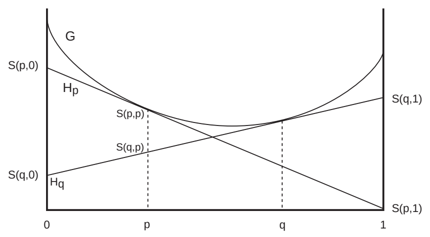


Figure 1: Illustration of a proper scoring rule. If the expert reports p, her expected score (relative to her true beliefs) is given by the subtangent hyperplane H_p. For any report q different from p, the expected score $S(q,p) = H_q \cdot p$ must be less than the expected score $S(p,p) = H_p \cdot p$ of truthful reporting.

for some convex G and subgradient G^*. S is strictly proper iff G is strictly convex.

Intuitively, Eq. 2 defines a hyperplane

$$H_{\mathbf{p}} = \langle S(\mathbf{p}, x_1), \ldots, S(\mathbf{p}, x_m) \rangle,$$

for each point $\mathbf{p}$, that is subtangent to G at $\mathbf{p}$. This defines a linear function, for any fixed report $\mathbf{p}$, giving the expected score of that report given beliefs $\mathbf{q}$: $S(\mathbf{p}, \mathbf{q}) = H_{\mathbf{p}} \cdot \mathbf{q}$. An illustration is given in Fig. 1 for a simple one-dimensional (two-outcome) scenario.

Several prediction market mechanisms allow the principal to extract information from multiple experts [18, 4]. *Market scoring rules (MSRs)* [11] allow experts to (sequentially) change the forecasted $\mathbf{p}$ using any proper S. An expert can change a forecast $\mathbf{p}'$ to $\mathbf{p}$ if she is willing to pay according to $S(\mathbf{p}', \cdot)$ and receive payment $S(\mathbf{p}, \cdot)$. If her true beliefs $\mathbf{p}$ differ from $\mathbf{p}'$ and the rule is strictly proper, then she has incentive to participate and report truthfully. Under certain conditions, MSRs can be interpreted as automated market makers [3]. Since each expert pays the amount due to the previous expert for her prediction, the net payment of the principal is the score associated with the final prediction.

Some prior work has studied incentives when prediction markets are used for decision making. Hanson [10] introduced the term *decision markets* to refer to the broad notion of prediction markets where experts offer forecasts for events conditional on some policy being adopted or a decision being taken. Osband [14] describes an interesting model for optimally incentivizing experts to "put effort" into deriving their forecasts. Othman and Sandholm [15] provide the first explicit, formal treatment of a principal who makes decisions based on expert forecasts. They address several key difficulties that arise due to the conditional nature of forecasts, but assume that the experts themselves have no direct interest in the decision that is taken. Chen and Kash [2, 6] extend this model to a wider class of informational settings and decision policies. Dimitrov and Sami [8] consider strategic behavior across multiple markets, where an expert may misreport her beliefs in one market to manipulate prices (and gain advantage) in another. Similarly, Conitzer [7] explores strategic aspects of prediction markets via connections to mechanism design, but again assumes that expert utility is derived solely from the mechanism's payoff. Shi et al. [17] consider experts that, once they report their forecasts, can take action to alter the probabilities of the outcomes in question. Unlike our model, they do not consider expert utility apart from the payoff offered by the mechanism (though, as in our model, the principal's utility function dictates the value of an expert report).

Hanson and Oprea [12] explicitly consider a single expert who has an interest in the final forecast of a prediction market and show that attempts to manipulate can in fact increase market accuracy (by incentivizing others to participate). Chen *et al.* [5] consider perfect Bayesian equilibria in a two-stage game with two participants predicting a boolean variable, one of whom has an interest in the final forecast. Both models, like ours, are motivated by expert interest in the principal's decision, but there are many important distinctions. Unlike our framework, neither approach explicitly models the principal's policy, utility, or loss due to manipulation. Both represent expert utility for the principal's decision indirectly through (very restrictive) payoff functions over the final forecast (either quadratic payoffs [12] or increasing payoffs [5]). Finally, neither model attempts to incentivize truthful reports by the manipulator.

3. SELF-INTERESTED EXPERTS

We now consider an expert who has a direct interest in the decision induced by her forecast. We devise a class of scoring rules that incentive self-interested agents to report their true beliefs. Such models are especially relevant in settings where expert opinions are sought from members internal to an organization. Rather than rejecting the forecasts of such experts, our model quantifies the impact of this self-interest and admits rules to circumvent it.

3.1 Model Formulation

A principal, or *decision maker (DM)*, elicits a forecast of $\mathcal{X}$ from expert E, and makes a decision based on this forecast. Let $D = \{d_1, \ldots, d_n\}$ be the set of possible decisions, and u_{ij} be DM's utility should he take decision d_i with x_j ($j \leq m$) being the realization of $\mathcal{X}$. Letting $\mathbf{u}_i = \langle u_{i1}, \cdots, u_{im}\rangle$, the expected utility of decision d_i given distribution $\mathbf{p}$ is $U_i(\mathbf{p}) = \mathbf{u}_i \cdot \mathbf{p}$. For any beliefs $\mathbf{p}$, DM takes the decision that maximizes expected utility, giving DM the *utility function* $U(\mathbf{p}) = \max_{i \leq n} U_i(\mathbf{p})$. Since each U_i is a linear function of $\mathbf{p}$, U is piecewise linear and convex (PWLC). Furthermore, each d_i is optimal in a (possibly empty) convex region of belief space $D_i = \{\mathbf{p} : \mathbf{u}_i \cdot \mathbf{p} \geq \mathbf{u}_j \cdot \mathbf{p}, \forall j\}$. We assume DM acts optimally and that he has a policy $\pi : \Delta(X) \rightarrow D$ that selects some optimal decision $\pi(\mathbf{p})$ for any expert forecast $\mathbf{p}$. In what follows, we take $D_i = \pi^{-1}(d_i)$. We denote by D_{ij} the (possibly empty) *boundary* between D_i and D_j. Notice that for any $\mathbf{p} \in D_{ij}$ we must have $U_i(\mathbf{p}) = U_j(\mathbf{p})$. In our running example, the CSO is given a forecast probability p of *high product demand* from the division head: he will authorize R&D if p is above some threshold τ, and abandon development if p falls below τ (here τ is the indifference probability: $U_{\text{develop}}(\tau) = U_{\text{abandon}}(\tau)$).

Our model of DM utility is slightly more restricted than that of [15, 2], who allow the utility of each decision to depend on a different random variable, and assume that a variable will be observed *only if* the corresponding decision is taken. We also focus on principals that maximize expected utility given E's report (i.e., DM uses the *max decision rule* [15]), though we discuss stochastic DM policies in Sec. 4.2.

Suppose expert E is asked by DM to provide a forecast of $\mathcal{X}$. Assume that E knows DM's policy π—knowledge of DM's utility function is sufficient but not required, see Sec. 4.1—and that E has her own utility function or *bias* b, where $b_{i,j}$ is E's utility should DM take decision d_i and x_j is the realization of $\mathcal{X}$. Define $\mathbf{b}_i = \langle b_{i,1}, \cdots, b_{i,m}\rangle$; and let E's expected utility for d_i given $\mathbf{p}$ be $B_i(\mathbf{p}) = \mathbf{b}_i \cdot \mathbf{p}$. For example, the division head (expert) may see increased corporate influence if R& D is authorized, but see her power wane if the product fails to materialize.

As with DM, E's *optimal utility function* B^* is PWLC:

$$B^*(\mathbf{p}) = \max_i \mathbf{b}_i \cdot \mathbf{p}. \tag{3}$$

Denote by $D^*(\mathbf{p})$ the decision d_i that maximizes Eq. 3, i.e., E's preferred decision given beliefs $\mathbf{p}$ (see Fig. 2).[2]

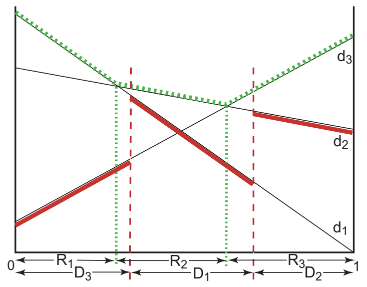


Figure 2: Expert utility function. E's utility for each decision d_i is given by the corresponding hyperplane (here, thin black line). E's "optimal" utility B^* is the PWLC function shown by the dotted green line (i.e., the upper surface), with each R_i denoting the regions of belief space where $D^*(\mathbf{p}) = d_i$. Regions D_i represent DM's policy, where $\pi(\mathbf{p}) = d_i$ for $\mathbf{p} \in D_i$. The thick red lines denote the (discontinuous) utility B^π that E will receive from (truthfully) reporting her belief.

Since DM is pursuing his own policy π, E's actual utility for a specific report $\mathbf{r}$ under beliefs $\mathbf{p}$ is given by

$$B^\pi(\mathbf{r}, \mathbf{p}) = \mathbf{b}_{\pi(\mathbf{r})} \cdot \mathbf{p}; \tag{4}$$

that is, if she reports $\mathbf{r}$, DM will take decision $\pi(\mathbf{r}) = d_k$ for some k, and she will derive benefit $B_k(\mathbf{p})$. We refer to $B^\pi(\mathbf{r}, \mathbf{p})$ as E's *inherent utility* for reporting $\mathbf{r}$. Similarly, $B(\mathbf{r}, x_i) = b_{i,\pi(\mathbf{r})}$ is E's inherent utility for report $\mathbf{r}$ under realization x_i. This is the inherent benefit she derives from the decision she induces DM to take. This is illustrated in Fig. 2. Note that E's utility for reports, given any fixed beliefs $\mathbf{p}$, is not generally continuous, with potential (jump) discontinuities at DM's decision boundaries.

Without some scoring rule, there is a clear incentive for E to misreport her true beliefs to induce DM to take a decision that E prefers, thereby causing DM to take a suboptimal decision. For instance, in Fig. 2, if E's true beliefs $\mathbf{p}$ lie in R_1, her preferred decision is d_1; but truthful reporting will induce DM to take decision d_3. E has greater inherent utility for reporting (any) $\mathbf{r} \in D_1$. Indeed, her gain from the misreport is $\mathbf{p} \cdot (\mathbf{b}_1 - \mathbf{b}_3)$. Equivalently, E stands to lose $\mathbf{p} \cdot (\mathbf{b}_1 - \mathbf{b}_3)$ by reporting truthfully. Intuitively, a proper scoring rule would remove this incentive to misreport.

3.2 Compensation Rules

If DM knows E utility function, he could reason about E's incentive to misreport and revise his decision policy accordingly. Of course, this would naturally lead to a Bayesian game requiring analysis of its Bayes-Nash equilibria, and generally leaving DM with uncertainty about E's true beliefs.[3] Instead, we wish to derive a scoring rule that DM can use to incentivize E to report truthfully.

Unsurprisingly, such rules must compensate E for the utility she foregoes by reporting her beliefs truthfully rather than influencing DM to act in a way that furthers E's own interests. A *compensation function* $C : \Delta(X) \times X \rightarrow \mathbb{R}$ maps reports and outcomes into payoffs, like a standard scoring rule. However, C does not fully determine E's utility for a report; we must also account for the inherent utility E derives from the decision she brings about. A compensation function C induces a *net scoring function*:

$$S(\mathbf{p}, x_i) = C(\mathbf{p}, x_i) + B^\pi(\mathbf{p}, x_i) \tag{5}$$

[2]We assume that D includes no dominated decisions; i.e., for any $d \in D$, we have $\pi^{-1}(d) \neq \emptyset$. If not, the subset of D with only nondominated decisions should be used, since DM never needs to compensate E for a decision he would never take.

[3]See Dimitrov and Sami [8] and Conitzer [7] for such a game-theoretic treatment of prediction markets (without decisions).

E's expected net score for report $\mathbf{r}$ under beliefs $\mathbf{p}$ is $S(\mathbf{r},\mathbf{p}) = C(\mathbf{r},\mathbf{p}) + B^\pi(\mathbf{r},\mathbf{p})$, where $C(\mathbf{r},\mathbf{p}) = \sum_i p_i C(\mathbf{r}, x_i)$ is E's expected compensation.

We adapt the definition of *proper* scoring rules to the case of compensation rules, recognizing that compensation is in full control of DM, while the net score is not:

DEFINITION 2. *A compensation function C is* proper *iff the expected net score function S satisfies $S(\mathbf{p},\mathbf{p}) \geq S(\mathbf{q},\mathbf{p})$ for all $\mathbf{p},\mathbf{q} \in \Delta(X)$. C is* strictly proper *if the inequality is strict.*

One natural way to structure the compensation function is to use C to compensate E for the loss in inherent utility incurred by reporting her true beliefs $\mathbf{p}$ (relative to her best report), thus removing incentive for E to misreport. This gives rise to a very specific compensation function C_b that accounts for this loss:

$$C_b(\mathbf{p}, x_i) = b_{i,D^*(\mathbf{p})} - b_{i,\pi(\mathbf{p})}. \tag{6}$$

$C_b(\mathbf{p}, x_i)$ is simply the difference between E's *realized* utility for her optimal decision (relative to her report $\mathbf{p}$) and the actual decision she induced. C_b gives rise to the specific net scoring function:

$$\begin{aligned} S_b(\mathbf{p}, x_i) &= C_b(\mathbf{p}, x_i) + B^\pi(\mathbf{p}, x_i) & (7)\\ &= (b_{i,D^*(\mathbf{p})} - b_{i,\pi(\mathbf{p})}) + b_{i,\pi(\mathbf{p})}) = b_{i,D^*(\mathbf{p})} & (8)\end{aligned}$$

Since E's expected net score under beliefs $\mathbf{p}$ is identical to her expected utility for the optimal decision $D^*(\mathbf{p})$, truthful reporting results, showing C_b to be a proper compensation rule.

C_b is just one straightforward mechanism for proper scoring with self-interested experts. We can generalize the approach to provide a complete characterization of all proper (and strictly proper) compensation functions. We derived C_b by compensating E for her *loss* due to truthful reporting. This is more "generous" than necessary: we need only *remove the potential gain* from misreporting. The key element of C_b is not the "compensation term" $b_{i,D^*(\mathbf{p})}$, but the penalty term $-b_{i,\pi(\mathbf{p})}$, which prevents E from benefiting by changing DM's decision. Any such gain is subtracted from her compensation via the penalty term $-b_{i,\pi(\mathbf{p})}$. We require only that the positive compensation term is convex: it need bear no connection to E's actual utility function to incentivize truthfulness. Indeed, we can fully characterize the space of proper and strictly proper compensation functions:

THEOREM 3. *A compensation rule C is proper for E iff*

$$C(\mathbf{p}, x_i) = G(\mathbf{p}) - G^*(\mathbf{p}) \cdot \mathbf{p} + G^*_i(\mathbf{p}) - b_{i,\pi(\mathbf{p})} \tag{9}$$

for some convex function G, and subgradient G^* of G. C is strictly proper iff G is strictly convex.[4]

An illustration of a cost function $G(\mathbf{p})$ that gives rise to a proper compensation function is shown in Fig. 3(a).

The fact that the specific rule C_b is proper follows directly by observing that the net score S_b can be derived from Eq. 2 by letting $G(\mathbf{p}) = B^*(\mathbf{p}) = \max_{i \leq n} B_i(\mathbf{p})$ be E's optimal utility function (which is PWLC, hence convex), and using the subgradient $G^*(\mathbf{p})$ given by the hyperplane corresponding to the optimal decision $D^*(\mathbf{p})$ at that point.[5] Of course, C_b is not *strictly* proper, since it is induced by a non-strictly convex cost function $G = B^*$. In particular, for any region $R(d)$ of belief space where a single decision d is optimal for E, every report $\mathbf{p} \in R(d)$ has the same expected net score, hence there is no "positive" incentive for truthtelling.

[4]All proofs are available in working paper *arXiv:1106.2489*.

[5]At interior points of E's decision regions, the hyperplane is the unique subgradient. At E's decision boundaries, an arbitrary subgradient can be used.

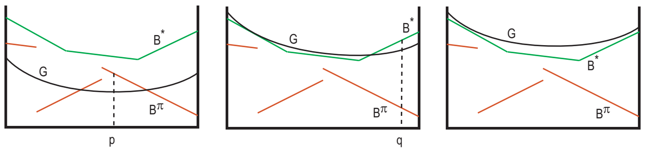


Figure 3: Illustration of cost functions G for strictly proper compensation rules. E's optimal utility B^* is the PWLC function shown in green, and E's inherent utility B^π is the discontinuous function in red. The net scoring function $G(\mathbf{p}) = S(\mathbf{p},\mathbf{p})$, the convex curve, induces an expected compensation function $C(\mathbf{p},\mathbf{p})$ by subtracting B^π. (a) A strictly proper rule that violates weak participation at point p. (b) A rule that satisfies weak participation but violates strong participation at point q. (c) A rule that satisfies strong participation.

The characterization of Thm. 3 ensures truthful reporting, but may not provide incentives for participation. Indeed, the expert may be *forced to pay DM in expectation* for certain beliefs. Specifically, if $G(\mathbf{p}) < B^\pi(\mathbf{p})$, E's expected compensation $C(\mathbf{p},\mathbf{p})$ is negative. Unless the DM can "force" E to participate, this will cause E to avoid providing a forecast if her beliefs are $\mathbf{p}$ (e.g., see point p is Fig. 3(a)). In general, we'd like to provide E with non-negative expected compensation. We can do this by insisting the compensation rule weakly incentives participation:

DEFINITION 4. *A compensation function C satisfies* weak participation *iff for any beliefs $\mathbf{p}$, E's expected compensation for truthful reporting $C(\mathbf{p},\mathbf{p})$ is non-negative.*

(Fig. 3(b) illustrates a cost function G that induces a compensation rule C satisfying weak participation.)

THEOREM 5. *A proper compensation rule C satisfies weak participation iff it meets the conditions of Thm. 3 and $G(\mathbf{p}) \geq B^\pi(\mathbf{p})$ for all $\mathbf{p} \in \Delta(X)$.*

While desirable, weak participation does not ensure participation in general. Consider a compensation function defined with a convex cost function $G(\mathbf{p})$. If E participates, she maximizes her net payoff by reporting her true beliefs $\mathbf{p}$. But suppose $G(\mathbf{p}) < B^*(\mathbf{p})$. While E may not be certain how DM will act without her input (e.g., she may not know DM's "default beliefs" precisely), she may nevertheless have beliefs about DM's default policy. And, if E believes DM will take decision $D^*(\mathbf{p})$ if she provides no forecast, then she is better off taking the expected payoff $B^*(\mathbf{p})$ based on her inherent utility and not participating (which has a lower payoff of $G(\mathbf{p})$). (See point q in Fig. 3(b).) To prevent this, we say C *strongly incentivizes* participation if, no matter what E believes about DM's default policy (i.e., his action given no reporting), she will not sacrifice expected utility by participating in the mechanism.

DEFINITION 6. *A compensation function C satisfies* strong participation *iff, for any decision $d_i \in D$, for any beliefs $\mathbf{p}$, E's net score for truthful reporting is no less than $B_i(\mathbf{p})$.*

Strong participation means that E has no incentive to abstain from participation (and need not "take her chances" that DM will make a decision she likes). This definition is equivalent to requiring that E's expected utility for truthful reporting, as a function of $\mathbf{p}$, is at least as great as her optimal utility function, i.e., $S(\mathbf{p},\mathbf{p}) \geq B^*(\mathbf{p})$ for all $\mathbf{p} \in \Delta(X)$. Fig. 3(c) illustrates such a compensation rule.

THEOREM 7. *Proper compensation rule C satisfies strong participation iff it meets the conditions of Thm. 3 and $G(\mathbf{p}) \geq B^*(\mathbf{p})$ for all $\mathbf{p} \in \Delta(X)$.*

OBSERVATION 8. *Compensation rule C_b is the unique minimal (non-strictly) proper rule satisfying strong participation. That is, no compensation rule offers lower compensation for any report without violating strong participation.*

In general, if we insist on strong participation, DM must provide potential compensation up to the level of E's maximum utility *gap*:

$$g(B) = \max_{i \le m, j, k \le n} b_{i,k} - b_{i,j}.$$

However, this degree of compensation is needed only if DM and E have "directly conflicting" interests (i.e., DM takes a decision whose realized utility is as far from optimal as possible from E's perspective). In such cases, one would expect E's utility to be significantly less than DM's. If not, this compensation is not worthwhile for DM. Conversely, if E's interests are well aligned with those of DM, the total compensation required will be small. The most extreme case of well-aligned utility is one where functions π and D^* coincide, i.e., $\pi(\mathbf{p}) = D^*(\mathbf{p})$ for all beliefs $\mathbf{p}$, in which case, no compensation is required. Specifically, compensation function $C_b(\mathbf{p}) = 0$ for all $\mathbf{p}$; and while C_b is not strictly proper, the only misreports that E will contemplate (i.e., that do not reduce her net score) are those that cannot change DM's decision (i.e., cannot impact DM's utility). As a consequence, DM should elicit forecasts from an expert who either (a) has well-aligned interests in the decisions being contemplated; (b) has interest whose magnitude is small (hence requires modest compensation) relative to DM's own utility; or (c) can be "forced" to make a prediction (possibly at negative *net* cost). Fortunately, these conditions often obtain in many settings, especially organizational or corporate settings. Employee incentives are usually reasonably well-aligned with those of corporate decision makers; and when external consultants are used, while their interests are not aligned with those of the principal, their stake in specific decisions is usually minimal.

4. POLICY AND UTILITY UNCERTAINTY

We now relax two key assumptions from Section 3.1: that E knows DM's policy, and that DM knows E's utility.

4.1 Policy Uncertainty

We first consider the case where DM does not want to disclose his policy to E. For example, suppose DM wanted to forego a truthful compensation rule C and simply rely on a proper scoring rule of the usual form that ignores the E's inherent utility. Thm. 3 shows that DM cannot prevent misreporting in general if he ignores E's inherent utility; hence he can suffer a loss in his own utility. However, by refusing to disclose his policy π, DM could reduce the incentive for E to misreport. Without accurate knowledge of π, E would be forced to rely on uncertain beliefs about π to determine the utility of a misreport, generally lowering her incentive. However, this will not remove the misreporting incentive completely. For instance, referring to Fig. 2, suppose DM does not disclose π. If E believes *with sufficient probability* that the decision boundary between d_3 and d_1 is located at the point indicated, she will misreport any forecast $\mathbf{p}$ in region D_3 sufficiently close to that boundary should DM use a scoring rule rather than a compensation rule. As such, refusing to disclose his policy can be used by DM to reduce, but not eliminate, the incentive to misreport.[6]

Our analysis in the previous section assumed that E used her knowledge of π to determine the report that maximizes her net score. However, DM does not need to disclose π to make good use of a compensation rule. He can specify a compensation rule *implicitly* by announcing his net scoring function $S(\mathbf{p}, x_i)$ (or the cost function G and subgradient G^*) and promising to deduct $B_d \cdot \mathbf{p}$ from this score for whatever decision d he ultimately takes. E *need not know* in advance what decision will be taken to be incentivized to offer a truthful forecast. Nor does E ever need to know what decisions *would have been taken* had she reported differently. Thus the only information E needs to learn about π is the value of $\pi(\mathbf{p})$ at her reported forecast $\mathbf{p}$; and even this need not be revealed until after the decision is taken (and its outcome realized).[7]

4.2 Uncertainty in Expert Utility

We now consider the more interesting issues that arise when DM is uncertain about the parameters $\mathbf{b}$ of E's utility function. If the DM has a distribution over $\mathbf{b}$, one obvious technique is to specify a proper compensation rule using the expectation of $\mathbf{b}$. This may work reasonably well in practice, depending on the nature of the distribution; but it follows immediately from Thm. 3 that this approach will not induce truthful reporting in general.

Rather than probabilistic beliefs, we suppose that DM has *constraints* on $\mathbf{b}$ that define a bounded feasible region $\mathcal{B} \subseteq \mathbb{R}^{mn}$ in which E's utility parameters must lie. We will confine our analysis to a simple, but natural class of constraints, specifically, upper and lower bounds on each utility parameter; i.e., assume DM has upper and lower bounds $b_{i,j}\!\uparrow$ and $b_{i,j}\!\downarrow$, respectively, on each $b_{i,j}$. This induces a hyper-rectangular feasible region $\mathcal{B}$. If $\mathcal{B}$ is a more general region (e.g., a polytope defined by more general linear constraints), our analysis below can be applied to the tightest "bounding box" of the feasible region.[8] Again by Thm. 3, DM cannot define a proper compensation rule in general: without certain knowledge of E's utility, any proposed "deduction" of inherent utility from E's compensation could mistaken, leading to an incentive to misreport. However, we show this incentive is bounded.

Requiring that DM have some information about E's utility for DM's decisions may, at first glance, seem like too stringent a requirement. However, in many contexts, including organizational settings like those discussed above, it would, in fact, be highly unusual for this *not* to be the case. For instance, it would be unheard of for a CSO not to have some rough, albeit imprecise, idea of the benefit a division head would derive from undertaking R&D for a new product. More to the point, ignoring E's potential biases makes it impossible for DM to have any confidence in her forecast. Our analysis sheds light on how much, and what type of, effort DM should invest in assessing E's biases.

Under conditions of utility uncertainty, it is natural for DM to restrict his attention to "consistent" compensation rules:

DEFINITION 9. *Let $\mathcal{B}$ be the set of feasible expert utility functions. A compensation rule is* consistent *with $\mathcal{B}$ iff it has the form, for some (strictly) convex G and $\tilde{\mathbf{b}} \in \mathcal{B}$:*

$$C(\mathbf{p}, x_i) = G(\mathbf{p}) - G^*(\mathbf{p}) \cdot \mathbf{p} + G_i^*(\mathbf{p}) - \tilde{b}_{i,\pi(\mathbf{p})}. \qquad (10)$$

Notice that consistent compensation rules are naturally linear: intuitively, we select a single consistent estimate of each parameter $\tilde{b_{i,j}} \in [b_{i,j}\!\downarrow, b_{i,j}\!\uparrow]$, treat E as if this were her true (linear) utility function, and define C using this estimate. We say DM is δ-*certain* of E's utility iff $b_{i,j}\!\uparrow - b_{i,j}\!\downarrow \le \delta$ for all i, j. Then we can bound the incentive for E to misreport as follows:

[6]A similar argument shows that a stochastic policy can be used to reduce misreporting incentive, e.g., the *soft max* policy that sees DM take decision d_i with probability proportional to $e^{\lambda u_i(\mathbf{p})}$. .

[7]Some mechanism to *verify* the decision *post hoc* may be needed.

[8]General linear constraints on E's parameters could be could be inferred, for example, from observed behavior.

THEOREM 10. *If DM is δ-certain of E's utility, then E's incentive to misreport under any consistent compensation rule is bounded by 2δ. That is, $S(\mathbf{r},\mathbf{p}) - S(\mathbf{p},\mathbf{p}) \leq 2\delta$.*

We can limit the misreporting incentive further by using a *uniform* compensation rule.

DEFINITION 11. *A consistent compensation rule is* uniform *if each parameter is estimated by $\tilde{b}_{i,\pi(\mathbf{p})} = \lambda b_{i,j}\!\downarrow + (1-\lambda) b_{i,j}\!\uparrow$ for some fixed $\lambda \in [0,1]$.*

For example, if DM uses the lower bound (or midpoint, or upper bound, etc.) of each parameter interval uniformly, we call the compensation rule uniform.

COROLLARY 12. *If DM is δ-certain of E's utility, then E's incentive to misreport under any uniform compensation rule is bounded by δ. That is, $S(\mathbf{r},\mathbf{p}) - S(\mathbf{p},\mathbf{p}) \leq \delta$.*

While bounding the *incentive* to misreport is useful, it is more important to understand the impact such misreporting can have on DM. Fortunately, this too can be bounded. The (strict) convexity of G means that the greatest incentive to misreport occurs at the decision boundaries of DM's policy π in Thm. 10. Since, by definition, DM is *indifferent* between the adjacent decisions at any decision boundary, misreports in a bounded region around decision boundaries have limited impact on DM's utility; the *amount* by which E will misreport is bounded using the "degree of convexity" of the cost function G, which in turn bounds DM's utility loss.

DEFINITION 13. *Let G be a convex cost function with subgradient G^*. We say G is* robust relative to G^* with factor $m > 0$ *iff, for all $\mathbf{p},\mathbf{q} \in \Delta(X)$:*[9]

$$G(\mathbf{q}) \geq G(\mathbf{p}) + G^*(\mathbf{p}) \cdot (\mathbf{q} - \mathbf{p}) + m||\mathbf{q} - \mathbf{p}||_2 \qquad (11)$$

It is not hard to see that m-robustness of the pair G, G^* imposes a minimum "penalty" on any expert misreport, as a function of its distance from her true beliefs:

OBSERVATION 14. *Let C be a proper compensation rule based on an m-robust cost function G and subgradient G^*. Let S be the induced net scoring function. Then*

$$S(\mathbf{p},\mathbf{p}) - S(\mathbf{q},\mathbf{p}) \geq m||\mathbf{q} - \mathbf{p}||_2.$$

Together with Thm. 10, this gives a bound on the degree to which an expert will misreport when an uncertain DM uses a consistent compensation rule.

COROLLARY 15. *Let DM be δ-certain of E's utility and use a consistent compensation rule based on an m-robust cost function and subgradient. Let $\mathbf{p}$ be E's true beliefs. Then the report $\mathbf{q}$ that maximizes E's net score satisfies $||\mathbf{q} - \mathbf{p}||_2 \leq \frac{2\delta}{m}$. If the compensation rule is uniform, then $||\mathbf{q} - \mathbf{p}||_2 \leq \frac{\delta}{m}$.*

In other words, E's utility-maximizing report must be within a bounded distance of her true beliefs if DM uses an m-robust cost function to define the compensation rule.

The notion of m-robustness is a slight variant of the notion of *strong convexity* [1] in which we use the specific subgradient G^* to measure the "degree of convexity." In the specific case of twice differentiable cost function G, we say G is *strongly convex with factor m* iff $\nabla^2 G(\mathbf{p}) \succeq mI$ for all $\mathbf{p} \in \Delta(X)$; i.e., if the matrix $\nabla^2 G(\mathbf{p}) - mI$ is positive definite [1]. m-convexity is a sufficient condition for the robustness we seek.

[9]The definition of m-robustness can be recast using any reasonable metric, e.g., L_1-norm or KL-divergence; but the L_2-norm is most convenient below when we relate robustness to strong convexity.

COROLLARY 16. *Let DM be δ-certain of E's utility and use a consistent compensation rule based on an m-convex, twice differentiable cost function G. Let $\mathbf{p}$ be E's true beliefs. Then the report $\mathbf{q}$ that maximizes E's net score satisfies $||\mathbf{q} - \mathbf{p}||_2 \leq \sqrt{\frac{4\delta}{m}}$. If the compensation rule is uniform, then $||\mathbf{q} - \mathbf{p}||_2 \leq \sqrt{\frac{2\delta}{m}}$.*

Robustness (or strong convexity) allows us to globally bound the maximum degree to which E will misreport. This allows us to give a simple, global bound on the loss in DM utility that results from his uncertainty about the expert's utility function. Recall that DM's utility function U_i for any decision d_i is linear, hence has a constant gradient ∇U_i. (We abuse notation and simply write ∇U_i for $\nabla U_i(\mathbf{p})$.) The function $U_i - U_j$ is also linear, given by parameter vector $(\mathbf{u}_i - \mathbf{u}_j)$. Let $\mathbf{e}_k$ denote the n-dimensional unit vector with a 1 in component k and zeros elsewhere.

THEOREM 17. *Let DM be δ-certain of E's utility and use a consistent compensation rule based on an m-robust cost function and subgradient. Assume E reports to maximize her net score. Then DM's loss in utility relative to a truthful report by E is at most $\max_k[\mathbf{e}_k^T \max_{i,j} \nabla(U_i - U_j)]\sqrt{n}\frac{2\delta}{m}$. If the compensation rule is uniform, then the bound is tightened by a factor of two.*

The same proof applies to strongly convex cost functions, with $\sqrt{n}\frac{2\delta}{m}$ replacing the term $\sqrt{n\frac{4\delta}{m}}$ in the bound above.

The results above all rely on the *global* robustness or *global* strong convexity of the cost function G. Designing a specific cost function (and if not differentiable, choosing its subgradients) can be challenging if we try to ensure uniform m-robustness or m-convexity across the entire probability space $\Delta(X)$. But recall that E can only impact DM's utility if her misreport causes DM to change his decision. This means that the cost function *need only induce strong penalties for misreporting near decision boundaries*. Furthermore, the strength of these penalties should be related to the rate at which DM's utility is negatively impacted. For example, suppose $\mathbf{p}$ lies on the decision boundary between region D_i and D_j. If $|\nabla(U_i - U_j)|$ is large, then a misreport in the region around $\mathbf{p}$ will cause a greater loss in utility than if $|\nabla(U_i - U_j)|$ is small. This suggests that the cost function should be more strongly convex (or more robust) near decision boundaries whose corresponding decisions differ significantly in utility, and can be less strong when the decisions are "similar." See Fig. 4 for an illustration of this point. Furthermore, the cost function need only be robust or strongly convex in a *local region* around these decision boundaries. In particular, suppose G is m-robust in some local region around the decision boundary between D_i and D_j. The degree of robustness bounds the maximum deviation from truth that E will contemplate. If the region of m-robustness includes these maximal deviations, that will be sufficient to bound DM's utility loss for any true beliefs E has in that region. Outside of these regions, no misreport by E will cause DM to change his decisions (relative to a truthful report).

We can summarize this as follows:

DEFINITION 18. *G is* locally robust relative to G^* *in the ε-neighborhood around $\mathbf{p}$ with factor $m > 0$ iff, for all $\mathbf{q} \in \Delta(X)$ s.t. $||\mathbf{q} - \mathbf{p}||_2 \leq \varepsilon$:*

$$G(\mathbf{q}) \geq G(\mathbf{p}) + G^*(\mathbf{p}) \cdot (\mathbf{q} - \mathbf{p}) + m||\mathbf{q} - \mathbf{p}||_2 \qquad (12)$$

Local strong convexity is defined similarly.

Now suppose DM wishes to bound his loss due to misreporting by E by some factor $\sigma > 0$. This can be accomplished using a locally robust cost function:

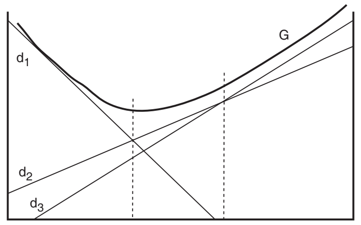


Figure 4: A locally strongly convex cost function G. Here G has is more strongly convex in the neighborhood of decision boundary D_{12} than the boundary D_{23}. This means an expert willing to sacrifice compensation (e.g., to gain inherent utility due to DM uncertainty) can offer a report that deviates more from her true beliefs in the neighborhood around D_{23} and than it can in the neighborhood around D_{12} for the same loss in compensation. However, since d_2 and d_3 are more similar than d_1 and d_2 (w.r.t. DM utility), i.e., the gradient $|\nabla(U_2 - U_3)|$ is less than $|\nabla(U_1 - U_2)|$, DM will lose less utility "per unit" of misreport in the neighborhood of D_{23}. Note: the cost function G is drawn above DM's utility function U for illustration only—in general, it will lie below U.

THEOREM 19. *Let DM be δ-certain of E's utility and fix $\sigma > 0$. For any pair of decisions d_i, d_j with non-empty decision boundary D_{ij}, define*

$$m_{ij} = \frac{\max_k(\mathbf{e}_k^T \nabla[U_i - U_j])\sqrt{n}2\delta}{\sigma}; \quad \varepsilon_{ij} = \frac{\sigma}{\max_k(\mathbf{e}_k^T \nabla[U_i - U_j])\sqrt{n}}.$$

Let G be a convex cost function with subgradient G^* such that, for all i, j and any $\mathbf{p} \in D_{ij}$, (a) G is locally robust with factor m_{ij} in the ε_{ij}-neighborhood around $\mathbf{p}$; (b) no other decision boundary lies within the ε_{ij}-neighborhood around $\mathbf{p}$. Let DM use a consistent compensation rule based on G, G^*. Assume E reports to maximize her net score. Then DM's loss in utility relative to a truthful report by E is at most σ. If the compensation rule is uniform, the result holds with m_{ij} and ε_{ij} decreased by a factor of two.

This result can be generalized to the case where the degree of robustness around one decision boundary is relaxed sufficiently so that the neighborhood within which E can profitably misreport crosses more than one decision boundary (i.e., when another decision boundary overlaps the ε_{ij}-neighborhood around D_{ij}). Utility loss will increase, but it can be bounded using the maximum gradient $\nabla(U_i - U_j)$ over decisions that can be swapped. The result can also be adapted to locally strongly convex cost functions.

These results quantify the "cost" to the decision maker of his imprecise knowledge of the expert's utility function, i.e., his worst-case expected utility relative to what he could have achieved if he had full knowledge of E's utility (i.e., with truthful reporting by E). This analysis, however, does more than merely bound the risk facing a principal who solicits forecasts from self-interested experts. It also suggests: (a) ways in which the principal might expend effort to refine his knowledge of expert utility or bias; and (b) procedures for optimizing compensation rules when dealing with such experts. Regarding the first issue, while it goes beyond the scope of this paper, a more fine-grained analysis in the style of that used here—based on DM uncertainty regarding E's *specific* utility parameters—can be used to justify and focus DM efforts when assessing E's utility. On issue (b), the characterization of DM loss using local robustness or convexity has operational significance in the design of compensation rules. Indeed, it suggests an procedure for designing cost function G (and induced compensation rule C) so as to minimize DM utility loss. Intuitively, G should optimize two conflicting objectives: minimizing the bound σ on utility loss (requiring an *increase* the degree of convexity at decision boundaries); and minimizing expected compensation c (requiring a *decrease* in convexity). We believe specific classes of spline functions should prove useful for addressing this tradeoff.

Finally, note that if we relax the constraint that DM choose the decision d_i with maximum expected utility, we can exploit local robustness to induce truthful forecasts. Suppose DM uses the softmax decision policy (see footnote 6): this stochastic policy makes E's utility $B^\pi(\mathbf{r}, \mathbf{p})$ continuous in her report $\mathbf{r}$. An similar analysis similar using local convexity shows that DM induces truthtelling if the degree of convexity compensates for the gradient of B^π at decision boundaries (since policy randomness removes the discontinuities in B^π). Of course, this comes at a cost: the DM is committed to taking suboptimal actions with some probability, leading to interesting tradeoffs between "acting optimally" but risking misleading reports vs. "acting suboptimally" given a truthful report.

5. MARKET SCORING RULES

We provide a brief sketch how to exploit compensation functions when DM aggregates the forecasts of multiple experts. One natural means of doing so is to use a *market scoring rule (MSR)* [11] that sequentially applies a scoring rule based on how an expert alters the prior forecast (see Sec. 2). An MSR based on a scoring rule S has the kth expert pay the k–1st expert for her forecast according to S, and have the principal pay only final expert for her forecast using S. Thus, the principal's total payment is bounded by the maximal payment to a single expert [11]. When experts are self-interested, however, difficulties emerge; e.g., Shi et al. [17] show that experts who can alter the outcome distribution after making a forecast *each* require compensation to prevent them from manipulating the distribution to the detriment to the principal. A related form of *subsidy* arises in our decision setting.

Following [17], we assume a collection of n experts, each of whom can provide alter the forecast $\mathbf{p}$ exactly once.[10] An "obvious" MSR in our model would simply adopt a proper compensation rule, and have each expert pay the either the *compensation* or the *net score* due to the expert who provided the incumbent forecast, and receive her payment from the next expert. If we use compensation, we run into strategic issues. With a proper compensation rule, an expert k reports truthfully based on her *net score (total utility)*, consisting of both compensation and the inherent utility of the decision she induces. In a market setting, k's proposed decision may be *changed* by the next expert's forecast. This (depending on her beliefs about other expert opinions) may incentivize k to misreport in order to *maximize her compensation* rather than her net score. Overcoming such strategic issues seems challenging.

Alternatively, each expert might pay the net score due her predecessor. Unfortunately, an arbitrary proper compensation rule may not pay expert k enough score to "cover her costs" (e.g., if k–1's inherent utility is much higher than k's). However, if we set aside issues associated with incentive for participation for the moment, the usual MSR approach can be adapted as follows: we fix a *single (strictly) convex cost function G for all experts*, and define the compensation rule C^k for expert k using G in the usual way:

$$C^k(\mathbf{p}, x_i) = G(\mathbf{p}) - G^*(\mathbf{p}) \cdot \mathbf{p} + G_i^*(\mathbf{p}) - b_{i,\pi(\mathbf{p})}^k,$$

where $\mathbf{b}^k$ is k's utility function (bias). If G satisfies strong participation for all experts (i.e., if $G(x_i) \geq B^*(x_i)$ for all i), then any

[10]This means we need not explicitly reason about how experts update their beliefs given the forecasts of others.

expert k whose beliefs $\mathbf{p}[k]$ differ from the forecast $\mathbf{p}[k\text{–}1]$ provided by k–1 will have an expected net score (given $\mathbf{p}[k]$) greater than her expected payment to k–1 and will maximize her utility by providing a truthful forecast. In particular, denote k's expected payment to k–1 by $\rho(k, k\text{–}1)$; then we have:

$$\begin{aligned}\rho(k, k\text{–}1) &= (H_{\mathbf{p}[k\text{–}1]} - \mathbf{b}^{k\text{–}1}_{\pi(\mathbf{p}[k\text{–}1])}) \cdot \mathbf{p}[k] \,+\, \mathbf{b}^{k\text{–}1}_{\pi(\mathbf{p}[k\text{–}1])} \cdot \mathbf{p}[k] \\ &= H_{\mathbf{p}[k\text{–}1]} \cdot \mathbf{p}[k] \\ &\leq H_{\mathbf{p}[k]} \cdot \mathbf{p}[k].\end{aligned}$$

Hence k's expected payment $\rho(k, k\text{–}1)$ is less than her expected net utility, leaving her with a (positive) net gain of $(H_{\mathbf{p}[k]} - H_{\mathbf{p}[k\text{–}1]}) \cdot \mathbf{p}[k]$. However, this gain may be smaller than the inherent utility she derives from the decision induced by k–1, namely, $\mathbf{b}^{k}_{\pi(\mathbf{p}[k\text{–}1])} \cdot \mathbf{p}[k]$. Hence this scheme may not incentivize participation. In cases where DM can force participation, such a scheme can be used; but in general, the self-subsidizing nature of standard MSRs *cannot* be exploited with self-interested experts.

To incentivize participation, DM can subsidize these payments. In the most extreme case, DM simply pays each displaced expert her net utility, which removes any incentives to misreport, but at potentially high cost. In certain circumstances, we can reduce the DM subsidy to the market by having him pay only the inherent utility $b^{k\text{–}1}_{i,\pi(\mathbf{p}[k\text{–}1])}$ (given realized outcome x_i) of the displaced expert k–1, and requiring the displacing expert k to pay the compensation $H_{i,\mathbf{p}[k\text{–}1]}$. Under certain conditions on the relative utility of different experts for different decisions, this is sufficient to induce participation; that is, k's net gain for participating exceeds her inherent utility for the incumbent decision.

For instance, suppose all experts have the same utility function $\mathbf{b}$ (e.g., consider experts in the same division of a company who are asked to predict the outcome of some event, and have different estimates, but have aligned interests in other respects). In this case, k's net gain for reporting her true beliefs is:

$$\begin{aligned}&(H_{\mathbf{p}[k]} - (H_{\mathbf{p}[k\text{–}1]} - \mathbf{b}_{\pi(\mathbf{p}[k\text{–}1])})) \cdot \mathbf{p}[k] \\ &\quad= (H_{\mathbf{p}[k]} - H_{\mathbf{p}[k\text{–}1]}) \cdot \mathbf{p}[k] + \mathbf{b}_{\pi(\mathbf{p}[k\text{–}1])} \cdot \mathbf{p}[k] \\ &\quad\geq \mathbf{b}_{\pi(\mathbf{p}[k\text{–}1])} \cdot \mathbf{p}[k].\end{aligned}$$

Hence k's expected net gain is at least as great as her inherent expected utility for the decision induced by k–1, and strictly greater if her beliefs differ from those of k–1. Thus participation is assured.

Indeed, the argument holds even if the utility functions are not identical: we require only that k's expected utility for the decision she displaces is less than the expected utility (given k's beliefs) to be offered to her predecessor k–1. A sufficient condition for this is that $\mathbf{b}^k \leq \mathbf{b}^{k\text{–}1}$ (pointwise). This suggests that if the DM can elicit predictions of the experts in a particular order, he should do so by eliciting forecasts of those with the greatest utility first. Even with identical expert utility functions, there seems to be no escape from the requirement that DM subsidize the market at a level that grows linearly with the number of agents (as in [17]). Further development of MSRs in this setting should prove to be quite interesting.

6. CONCLUDING REMARKS

We have presented a model for the analysis of the incentives facing experts who have a vested interest in the decision taken by the principal, defining *compensation rules* that are necessary and sufficient to induce truthful forecasts and ensure participation. The analysis allows for uncertainty in the knowledge of both parties, exploiting various forms of robustness or convexity. We also provided some initial steps toward MSRs based on compensation rules. Rather than rejecting self-interested experts outright, our model allows the principal to assess the risks of using such experts, and design compensation to mitigates these risks.

Of course, our model and analysis are just first steps toward a comprehensive treatment of self-interested experts. Many interesting directions remain, including: the development of effective procedures for the design of cost functions that minimize utility loss and compensation when expert utility is unknown; the design and analysis of more refined market-scoring rules; finer-grained modeling of DM utility loss to guide DM's elicitation or assessment effort of expert utility/interests; and analyzing the tradeoffs when DM accepts restrictions on his possible decisions (potentially acting suboptimally) to reduce expert misreporting.

Acknowledgements: Thanks to Yiling Chen, Vince Conitzer, Ian Kash, Tuomas Sandholm, Jenn Wortman Vaughan and the anonymous reviewers for helpful discussions and suggestions. This work was supported by NSERC.

7. REFERENCES

[1] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
[2] Y. Chen and I. Kash. Information elicitation for decision making. *AAMAS-11*, pp.175–182, Taipei, 2011.
[3] Y. Chen and D. Pennock. A utility framework for bounded-loss market makers. *UAI-07*, pp.49–56, 2007.
[4] Y. Chen and D. Pennock. Designing markets for prediction. *AI Magazine*, 31(4):42–52, 2011.
[5] Y. Chen, X. A. Gao, R. Goldstein, and I. A. Kash. Market manipulation with outside incentives. *AAAI-11*, pp.614–619, San Francisco, 2011.
[6] Y. Chen, I. A. Kash, M. Ruberry, and V. Shnayde. Decision markets with good incentives. *WINE-11*, pp.72–83, Singapore, 2011.
[7] V. Conitzer. Prediction markets, mechanism design, and cooperative game theory. *UAI-09*, pp.101–108, 2009.
[8] S. Dimitrov and R. Sami. Non-myopic strategies in prediction markets. *ACM EC'08*, pp.200–209, 2008.
[9] T. Gneiting and A. Raftery. Strictly proper scoring rules, prediction, and estimation. *J. Amer. Stat. Assoc.*, 102:359–378, 2007.
[10] R. Hanson. Decision markets. *IEEE Intelligent Systems*, 14:16–19, 1997.
[11] R. Hanson. Logarithmic market scoring rules for modular combinatorial information aggregation. *J. Prediction Markets*, 1:3–15, 2007.
[12] R. Hanson and R. Oprea. A manipulator can aid prediction market accuracy. *Economica*, 76(302):304–314, 2009.
[13] J. McCarthy. Measures of the value of information. *PNAS*, 42(9):654–655, 1956.
[14] K. Osband. Optimal forecasting incentives. *J. Pol. Econ.*, 97(5):1091–1112, 1989.
[15] A. Othman and T. Sandholm. Decision rules and decision markets. *AAMAS-10*, pp.625–632, Toronto, 2010.
[16] L. J. Savage. Elicitation of personal probabilities and expectations. *J. Amer. Stat. Assoc.*, 66:783–801, 1971.
[17] P. Shi, V. Conitzer, M. Guo. Prediction mechanisms that do not incentivize undesirable actions. *WINE-09*, pp.89–100, 2009.
[18] J. Wolfers and E. Zitzewitz. Prediction markets. *J. Econ. Perspectives*, 18(2):107–126, 2004.

Worst-Case Optimal Redistribution of VCG Payments in Heterogeneous-Item Auctions with Unit Demand

Mingyu Guo
Department of Computer Science
University of Liverpool, UK
Mingyu.Guo@liverpool.ac.uk

ABSTRACT

Many important problems in multiagent systems involve the allocation of multiple resources among the agents. For resource allocation problems, the well-known VCG mechanism satisfies a list of desired properties, including efficiency, strategy-proofness, individual rationality, and the non-deficit property. However, VCG is generally not budget-balanced. Under VCG, agents pay the VCG payments, which reduces social welfare. To offset the loss of social welfare due to the VCG payments, VCG redistribution mechanisms were introduced. These mechanisms aim to redistribute as much VCG payments back to the agents as possible, while maintaining the aforementioned desired properties of the VCG mechanism.

We continue the search for worst-case optimal VCG redistribution mechanisms – mechanisms that maximize the fraction of total VCG payment redistributed in the worst case. Previously, a worst-case optimal VCG redistribution mechanism (denoted by WCO) was characterized for multi-unit auctions with nonincreasing marginal values [7]. Later, WCO was generalized to settings involving heterogeneous items [4], resulting in the HETERO mechanism. [4] *conjectured* that HETERO is feasible and worst-case optimal for heterogeneous-item auctions with unit demand. In this paper, we propose a more natural way to generalize the WCO mechanism. We prove that our generalized mechanism, though represented differently, actually coincides with HETERO. Based on this new representation of HETERO, we prove that HETERO is indeed feasible and worst-case optimal in heterogeneous-item auctions with unit demand. Finally, we conjecture that HETERO remains feasible and worst-case optimal in the even more general setting of combinatorial auctions with gross substitutes.

Categories and Subject Descriptors

J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*; I.2.11 [**Computing Methodologies**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Economics, Theory

Keywords

Mechanism design, Vickrey-Clarke-Groves mechanism, payment redistribution

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

1. INTRODUCTION

1.1 VCG Redistribution Mechanisms

Many important problems in multiagent systems involve the allocation of multiple resources among the agents. For resource allocation problems, the well-known VCG mechanism satisfies the following list of desired properties:

- *Efficiency*: the allocation maximizes the agents' total valuation (without considering payments).
- *Strategy-proofness*: for any agent, reporting truthfully is a dominant strategy, regardless of the other agents' types.
- *(Ex post) individual rationality*: Every agent's final utility (after deducting her payment) is always nonnegative.
- *Non-deficit*: the total payment *from* the agents is nonnegative.

However, VCG is generally not budget-balanced. Under VCG, agents pay the VCG payments, which reduces social welfare. To offset the loss of social welfare due to the VCG payments, VCG redistribution mechanisms were introduced. These mechanisms still allocate the resources using VCG. On top of VCG, these mechanisms try to redistribute as much VCG payments back to the agents as possible. We require that *an agent's redistribution be independent of her own type*. This is sufficient for maintaining strategy-proofness and efficiency (an agent has no control over her own redistribution). For smoothly connected domains (including multi-unit auctions with nonincreasing marginal values and heterogeneous-item auctions with unit demand), the above requirement is also necessary for maintaining strategy-proofness and efficiency [8]. A VCG redistribution mechanism is *feasible* if it maintains all the desired properties of the VCG mechanism. That is, we also require that the redistribution process maintains individual rationality and the non-deficit property.

Let n be the number of agents. Since all VCG redistribution mechanisms start by allocating according to the VCG mechanism, a VCG redistribution mechanism is characterized by its redistribution scheme $\vec{r} = (r_1, r_2, \ldots, r_n)$. Under VCG redistribution mechanism $\vec{r}$, agent i's redistribution equals $r_i(\theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_n)$, where θ_j is agent j's type. (We do not have to differentiate between an agent's true type and her reported type, since all VCG redistribution mechanisms are strategy-proof.) For the mechanism design objective studied in this paper, it is without loss of generality to only consider VCG redistribution mechanisms that are anonymous (we defer the proof of this claim to the appendix). An anonymous VCG redistribution mechanism is characterized by a single function r. Under (anonymous) VCG redistribution mechanism r, agent i's redistribution equals $r(\theta_{-i})$, where θ_{-i} is the *multiset* of the types of the agents other than i.

We use $\vec{\theta}$ to denote the type profile. Let $VCG(\vec{\theta})$ be the total VCG payment for this type profile. A VCG redistribution mechanism r satisfies the non-deficit property if the total redistribution never exceeds the total VCG payment. That is, for any type profile $\vec{\theta}$, $\sum_i r(\theta_{-i}) \leq VCG(\vec{\theta})$. A VCG redistribution mechanism r is (ex post) individually rational if every agent's final utility is always nonnegative. Since VCG is individually rational, we have that a sufficient condition for r to be individually rational is for any $\vec{\theta}$ and any i, $r(\theta_{-i}) \geq 0$ (on top of VCG, every agent also receives a redistribution amount that is always nonnegative). On the other hand, when agent i is not interested in any item (her valuation on any item bundle equals 0), under VCG, i's utility always equals 0. After redistribution, agent i's utility is exactly her redistribution $r(\theta_{-i})$. That is, $r(\theta_{-i}) \geq 0$ for all θ_{-i} (hence for all $\vec{\theta}$ and all i) is also necessary for individual rationality.

We want to find VCG redistribution mechanisms that maximize *the fraction of total VCG payment redistributed in the worst-case*. This mechanism design problem is equivalent to the following functional optimization model:

Variable function: r
Maximize: α (worst-case redistribution fraction)
Subject to:
Non-deficit: $\forall \vec{\theta}$, $\sum_i r(\theta_{-i}) \leq VCG(\vec{\theta})$
Individual rationality: $\forall \theta_{-i}$, $r(\theta_{-i}) \geq 0$
Worst-case guarantee: $\forall \vec{\theta}$, $\sum_i r(\theta_{-i}) \geq \alpha VCG(\vec{\theta})$

In this paper, we will analytically characterize one worst-case optimal VCG redistribution mechanism for heterogeneous-item auctions with unit demand.[1]

We conclude this subsection with an example VCG redistribution mechanism in the simplest setting of single-item auctions. In a single-item auction, an agent's type is a nonnegative real number representing her utility for winning the item. Without loss of generality, we assume that $\theta_1 \geq \theta_2 \geq \ldots \geq \theta_n \geq 0$. In single-item auctions, the Bailey-Cavallo VCG redistribution mechanism [2, 3] works as follows:

- Allocate the item according to VCG: Agent 1 wins the item and pays θ_2. The other agents win nothing and do not pay.
- Every agent receives a redistribution that equals $\frac{1}{n}$ times the second highest *other* type: Agent 1 and 2 each receives $\frac{1}{n}\theta_3$. The other agents each receives $\frac{1}{n}\theta_2$.

The above mechanism obviously maintains strategy-proofness and efficiency (an agent's redistribution does not depend on her own type). It also maintains individual rationality because all redistributions are nonnegative. The total redistribution equals $\frac{2}{n}\theta_3 + \frac{n-2}{n}\theta_2$. This is never more than the total VCG payment θ_2. That is, the above mechanism maintains the non-deficit property. Finally, the total redistribution $\frac{2}{n}\theta_3 + \frac{n-2}{n}\theta_2 \geq \frac{n-2}{n}\theta_2$. That is, for single-item auctions, this example mechanism's worst-case redistribution fraction is $\frac{n-2}{n}$ (the worst-case is reached when $\theta_3 = 0$).

1.2 Previous Research on Worst-Case Optimal VCG Redistribution Mechanisms

In this subsection, we review existing results on worst-case optimal VCG redistribution mechanisms. Besides high-level discussions, we also choose to include a certain level of technical details, as they are needed for later sections.

We organize existing results by their settings.

Worst-Case Optimal Redistribution in Multi-Unit Auctions with Unit Demand [7, 12]: In multi-unit auctions with unit demand, the items for sale are identical. Each agent wants at most one copy of the item. (Single-item auctions are special cases of multi-unit auctions with unit demand.) Let m be the number of items. *Throughout this paper, we only consider cases where $m \leq n-2$.*[2] Here, an agent's type is a nonnegative real number representing her valuation for winning one copy of the item. It is without loss of generality to assume that $\theta_1 \geq \theta_2 \geq \ldots \geq \theta_n \geq 0$. [7] showed that for multi-unit auctions with unit demand, any VCG redistribution mechanism's worst-case redistribution fraction is at most

$$\alpha^* = 1 - \frac{\binom{n-1}{m}}{\sum_{j=m}^{n-1}\binom{n-1}{j}}$$

If we switch to a more general setting, then α^* is still an upper bound: if there exists a VCG redistribution mechanism whose worst-case redistribution fraction is strictly larger than α^* in a more general setting, then this mechanism, when applied to multi-unit auctions with unit demand, has a worst-case redistribution fraction that is strictly larger than α^*, which contradicts with the meaning of α^*.

[7] also characterized a VCG redistribution mechanism for multi-unit auctions with unit demand, called the WCO mechanism.[3] WCO's worst-case redistribution fraction is exactly α^*. That is, it is worst-case optimal.

WCO was obtained by optimizing within the family of *linear* VCG redistribution mechanisms. A linear VCG redistribution mechanism r takes the following form:

$$r(\theta_{-i}) = \sum_{j=1}^{n-1} c_j [\theta_{-i}]_j$$

Here, the c_i are constants. (We only consider the c_i that correspond to feasible VCG redistribution mechanisms.) $[\theta_{-i}]_j$ is the j-th highest type among θ_{-i}. Linear mechanism r is characterized by the values of the c_i. The optimal values the c_i are as follows:

$$c_i^* = \frac{(-1)^{i+m-1}(n-m)\binom{n-1}{m-1}}{i\sum_{j=m}^{n-1}\binom{n-1}{j}} \frac{1}{\binom{n-1}{i}} \sum_{j=i}^{n-1}\binom{n-1}{j}$$

for $i = m+1, \ldots, n-1$, and $c_i^* = 0$ for $i = 1, 2, \ldots, m$.

The characterization of WCO then follows:

$$r(\theta_{-i}) = \sum_{j=1}^{n-1} c_j^* [\theta_{-i}]_j = \sum_{j=m+1}^{n-1} c_j^* [\theta_{-i}]_j$$

Worst-Case Optimal Redistribution in Multi-Unit Auctions with Nonincreasing Marginal Values [7]: Multi-unit auctions with non-

[1]The problem of assigning heterogeneous items to unit demand agents is also often called the assignment problem.

[2][7] showed that for multi-unit auctions with unit demand, when $m = n-1$, the worst-case redistribution fraction (of any feasible VCG redistribution mechanism) is at most 0. Since the setting studied in this paper is more general (heterogeneous-item auctions with unit demand), we also have that the worst-case redistribution fraction is at most 0 when $m = n-1$. Since heterogeneous-item auctions with x units are special cases of heterogeneous-item auctions with $x+1$ units, we have that for our setting the worst-case redistribution fraction is at most 0 when $m \geq n-1$. That is, not redistributing anything is worst-case optimal when $m \geq n-1$.

[3]WCO has also been independently derived in [12], under a slightly different objective of maximizing worst-case efficiency ratio. Also, for [12]'s objective, the optimal mechanism coincides with WCO only when the individual rationality constraint is enforced.

increasing marginal values are more general than multi-unit auctions with unit demand. In this more general setting, the items are still identical, but an agent may demand more than one copy of the item. An agent's valuation for winning the first copy of the item is called her initial/first marginal value. Similarly, an agent's additional valuation for winning the i-th copy of the item is called her i-th marginal value. An agent's type contains m nonnegative real numbers (i-th marginal value for $i = 1, \ldots, m$). In this setting, it is further assumed that the marginal values are nonincreasing.

As discussed earlier, in this more general setting, any VCG redistribution mechanism's worst-case redistribution fraction is still bounded above by α^*. [7] generalized WCO to this setting, and proved that its worst-case redistribution fraction remains the same. Therefore, WCO (after generalization) is also worst-case optimal for multi-unit auctions with nonincreasing marginal values.

The original definition of WCO does not directly generalize to multi-unit auctions with nonincreasing marginal values. When it comes to multi-unit auctions with nonincreasing marginal values, an agent's type is no longer a single value, which means that there is no such thing as "the j-th highest type among θ_{-i}". To address this, [7] replaced $[\theta_{-i}]_j$ by $\frac{1}{m}R(\theta_{-i}, j-m-1)$ for $j = m+1, \ldots, n-1$. Basically, $R(\theta_{-i}, j-m-1)$ is the generalization of $[\theta_{-i}]_j$: it is identical to $[\theta_{-i}]_j$ in the unit demand setting, and it remains well-defined for multi-unit auctions with nonincreasing marginal values. We abuse notation by not differentiating the agents and their types. For example, θ_{-i} is equivalent to the set of agents other than i. Let S be a set of agents. $R(S, i)$ is formally defined as follows (this definition is included for completeness; we will not use it anywhere):

- $R(S, 0) = VCG(S)$ (the total VCG payment when only those in S participate in the auction).
- For $i = 1, \ldots, |S|-m-1$, $R(S, i) = \frac{1}{m+i}\sum_{j=1}^{m+i} R(U(S, j), i-1)$. Here, $U(S, j)$ is the new set of agents, after removing the agent with the j-th highest initial marginal value in S from S.

The general form of WCO is as follows:

$$r(\theta_{-i}) = \frac{1}{m}\sum_{j=m+1}^{n-1} c_j^* R(\theta_{-i}, j-m-1)$$

Worst-Case Optimal Redistribution in Heterogeneous-Item Auctions with Unit Demand [4]: In heterogeneous-item auctions with unit demand, the items for sale are different. Each agent demands at most one item. Here, an agent's type consists of m nonnegative real numbers (her valuation for winning item i for $i = 1, \ldots, m$). Heterogeneous-item auctions with unit demand is the main focus of this paper.

Since heterogeneous-item auctions with unit demand is more general than multi-unit auctions with unit demand, α^* is still an upper bound on the worst-case redistribution fraction. [4] proposed the HETERO mechanism, by generalizing WCO. The authors *conjectured* that HETERO is feasible and has a worst-case redistribution fraction that equals α^*. That is, the authors conjectured that HETERO is worst-case optimal in this setting. The main contribution of this paper is a proof of this conjecture.

Redistribution in Combinatorial Auctions with Gross Substitutes [6]: The gross substitutes condition was first proposed in [9]. Like unit demand, the gross substitutes condition is a condition on an agent's type (does not depend on the mechanism under discussion). In words, an agent's type satisfies the gross substitutes condition if her demand for an item does not decrease when the prices of the other items increase. Both multi-unit auctions with nonincreasing marginal values and heterogeneous-item auctions with unit demand are special cases of combinatorial auctions with gross substitutes [5, 9]. [6] showed that for this setting, the worst-case redistribution fraction of the Bailey-Cavallo mechanism [2, 3] is exactly $\frac{n-m-1}{n}$ (when $n \geq m+1$), and it is possible to construct mechanisms with even higher worst-case redistribution fractions. The authors did not find a worst-case optimal mechanism for this setting. At the end of this paper, we conjecture that HETERO is optimal for combinatorial auctions with gross substitutes.

Finally, Naroditskiy *et al.* [13] proposed a numerical technique for designing worst-case optimal redistribution mechanisms. The proposed technique only works for single-parameter domains. It does not apply to our setting (multi-parameter domain).

1.3 Our contribution

We generalize WCO to heterogeneous-item auctions with unit demand. We prove that the generalized mechanism, though represented differently, coincides with the HETERO mechanism proposed in [4]. That is, what we proposed is not a new mechanism, but a new representation of an existing mechanism. Based on our new representation of HETERO, we prove that HETERO is indeed feasible and worst-case optimal when applied to heterogeneous-item auctions with unit demand, thus confirming the conjecture raised in [4]. We conclude with a new conjecture that HETERO remains feasible and worst-case optimal in the even more general setting of combinatorial auctions with gross substitutes.

2. NEW REPRESENTATION OF HETERO

We recall that WCO was obtained by optimizing within the family of linear VCG redistribution mechanisms. The original representation of HETERO was obtained using a similar approach [4]. The authors focused on the following family of mechanisms:

$$r(\theta_{-i}) = \sum_{j=1}^{n-m-1} \beta_j t(\theta_{-i}, j-1)$$

Here, the β_i are constants. $t(S, j)$ is the *expected* total VCG payment when we remove j agents uniformly at random from S, and allocate all the items to the remaining agents. It is easy to see that all member mechanisms of the above family are well-defined for general combinatorial auctions. Not every member mechanism is feasible though.

[4] did not attempt optimizing over the family. Instead, the β_i are chosen so that the corresponding mechanism coincides with WCO when it comes to multi-unit auctions with unit demand. It turns out that the choice is *unique*, and the corresponding mechanism is called HETERO. [4] *conjectured* that HETERO is feasible and worst-case optimal for heterogeneous-item auctions with unit demand.

In this section, we propose another way to generalize WCO. We will show that the generalized WCO actually coincides with HETERO. That is, what we derive is a new representation of HETERO. This new representation will prove itself useful in later discussions.

We recall that the characterization of WCO for multi-unit auctions with nonincreasing marginal values is based on a series of functions $R(S, i)$. These functions do not directly generalize to settings involving heterogeneous items, because, for $i > 0$, $R(S, i)$ is defined explicitly based on the agents' initial marginal values. Fortunately, there is an easy way to rewrite $R(S, i)$, so that it becomes well-defined for settings involving heterogeneous items.

[7] proved that for $0 \leq j \leq |S| - m - 2$,

$$\sum_{a \in S} R(S-a, j) = (|S|-m-1-j)R(S,j)+(m+1+j)R(S,j+1) \quad (1)$$

Based on Equation 1, WCO can be rewritten into the following form (the only changes are that for $i > 0$, $R(S,i)$'s definition no longer mentions "initial marginal values"):

Definition 1. Heterogeneous WCO (new representation of HETERO):

$$r(\theta_{-i}) = \frac{1}{m} \sum_{j=m+1}^{n-1} c_j^* R(\theta_{-i}, j-m-1)$$

- $R(S,0) = VCG(S)$
- For $i = 1, \ldots, |S| - m - 1$, $R(S,i)$ equals:

$$\frac{1}{m+i}\left(\sum_{a \in S} R(S-a, i-1) - (|S| - m - i)R(S, i-1)\right)$$

Heterogeneous WCO is well-defined for general combinatorial auctions, so we can directly apply it to heterogeneous-item auctions with unit demand. Of course, we still have the burden to prove that it remains feasible and worst-case optimal. We will do so in the next section.

Heterogeneous WCO is not a new mechanism. It turns out that it coincides with HETERO for general combinatorial auctions. That is, Definition 1 is a new representation of the existing mechanism HETERO.

PROPOSITION 1. *Heterogeneous WCO coincides with HETERO for general combinatorial auctions.*

Proof omitted since it is based on pure algebraic manipulation.

3. FEASIBILITY AND WORST-CASE OPTIMALITY OF HETERO

In this section, we prove that HETERO, as represented in Definition 1, is feasible and worst-case optimal for heterogeneous-item auctions with unit demand.

We first define the *redistribution monotonicity* condition:

Definition 2. An auction setting satisfies *redistribution monotonicity* if for any set of agents S, we have that

$$R(S,0) \geq R(S,1) \geq \ldots \geq R(S, |S| - m - 1) \geq 0$$

R was defined in Definition 1. That is, $R(S,0) = VCG(S)$, and for $i = 1, \ldots, |S| - m - 1$, $R(S,i)$ equals

$$\frac{1}{m+i}\left(\sum_{a \in S} R(S-a, i-1) - (|S| - m - i)R(S, i-1)\right).$$

For example, the setting of single-item auctions satisfies redistribution monotonicity. In a single-item auction, $R(S,0) = VCG(S) = [S]_2$ ($[S]_i$ is the i-th highest type from the agents in S).

$$R(S,1) = \frac{1}{2}\left(\sum_{a \in S} R(S-a, 0) - (|S| - 2)R(S,0)\right)$$

$$= \frac{1}{2}\left(2[S]_3 + (|S|-2)[S]_2 - (|S|-2)[S]_2\right) = [S]_3.$$

Similarly, $R(S,2) = [S]_4$, $R(S,3) = [S]_5, \ldots$, and finally $R(S, |S| - m - 1) = R(S, |S| - 2) = [S]_{|S|}$ (lowest type from the agents in S). It is clear that redistribution monotonicity holds here.

More generally, redistribution monotonicity holds for multi-unit auctions with nonincreasing marginal values: Claim 17 of [7] proved that $R(S,i)$ is nonincreasing in i for multi-unit auctions with nonincreasing marginal values; $R(S,i)$'s original definition as described in Subsection 1.2 makes it clear that the $R(S,i)$ are nonnegative.

The following proposition greatly simplifies our task:

PROPOSITION 2. *If the setting satisfies redistribution monotonicity, then HETERO is feasible (strategy-proof, efficient, individually rational, and non-deficit), and its worst-case redistribution fraction is at least α^*. If the setting is also more general than multi-unit auctions with unit demand, then HETERO is worst-case optimal.*

PROOF. We first prove that HETERO is feasible given redistribution monotonicity. According to Definition 1, under HETERO, an agent's redistribution does not depend on her own type. That is, HETERO is strategy-proof and efficient in all settings. We only need to prove that HETERO is individually rational and non-deficit given redistribution monotonicity.

Individual rationality: As discussed in Subsection 1.1, individual rationality is equivalent to redistributions being nonnegative. We recall that for multi-unit auctions with unit demand, under WCO, agent i's redistribution equals

$$r(\theta_{-i}) = \sum_{j=m+1}^{n-1} c_j^* [\theta_{-i}]_j$$

WCO is known to be individually rational. That is, for all θ_{-i},

$$\sum_{j=m+1}^{n-1} c_j^* [\theta_{-i}]_j \geq 0$$

This is equivalent to for all $x_0 \geq \ldots \geq x_{n-m-2} \geq 0$,

$$\sum_{j=m+1}^{n-1} c_j^* x_{j-m-1} \geq 0 \quad (2)$$

Under HETERO, agent i's redistribution equals

$$\frac{1}{m} \sum_{j=m+1}^{n-1} c_j^* R(\theta_{-i}, j-m-1) \quad (3)$$

Redistribution monotonicity implies that

$$R(\theta_{-i}, 0) \geq R(\theta_{-i}, 1) \geq \ldots \geq R(\theta_{-i}, n-m-2) \geq 0 \quad (4)$$

Based on (2) and (4) (substituting $R(\theta_{-i}, j)$ for x_j for all j), we have that (3) is nonnegative. Therefore, redistribution monotonicity implies individual rationality.

Non-deficit and worst-case optimality: For multi-unit auctions with unit demand, under WCO, the total VCG payment is $m\theta_{m+1}$. The total redistribution is

$$\sum_{i=1}^{n} \sum_{j=m+1}^{n-1} c_j^* [\theta_{-i}]_j = \sum_{j=m+1}^{n-1} c_j^* \sum_{i=1}^{n} [\theta_{-i}]_j$$

$$= \sum_{j=m+1}^{n-1} c_j^* (j\theta_{j+1} + (n-j)\theta_j)$$

WCO is known to be non-deficit and have worst-case redistribution fraction α^*. That is, for all $\theta_{m+1} \geq \ldots \geq \theta_n \geq 0$,

$$\alpha^* m\theta_{m+1} \leq \sum_{j=m+1}^{n-1} c_j^* (j\theta_{j+1} + (n-j)\theta_j) \leq m\theta_{m+1}$$

That is, for all $x_0 \geq x_1 \geq \ldots \geq x_{n-m-1} \geq 0$,

$$\alpha^* m x_0 \leq \sum_{j=m+1}^{n-1} c_j^*(j x_{j-m} + (n-j) x_{j-m-1}) \leq m x_0 \quad (5)$$

Under HETERO, the total redistribution is

$$\frac{1}{m} \sum_{i=1}^{n} \sum_{j=m+1}^{n-1} c_j^* R(\theta_{-i}, j-m-1)$$

$$= \frac{1}{m} \sum_{j=m+1}^{n-1} c_j^*(jR(\vec{\theta}, j-m) + (n-j)R(\vec{\theta}, j-m-1)) \quad (6)$$

The total VCG payment equals $VCG(\vec{\theta}) = R(\vec{\theta}, 0)$. Redistribution monotonicity implies that

$$R(\vec{\theta}, 0) \geq R(\vec{\theta}, 1) \geq \ldots \geq R(\vec{\theta}, n-m-1) \geq 0 \quad (7)$$

Given (5) and (7) (substituting $R(\vec{\theta}, j)$ for x_j for all j), we have that (6) is between α^* times the total VCG payment and the total VCG payment. Therefore, redistribution monotonicity implies the non-deficit property and also worst-case optimality. □

In the remaining of this section, we prove that heterogeneous-item auctions with unit demand satisfies redistribution monotonicity, which would then imply that HETERO is feasible and worst-case optimal for heterogeneous-item auctions with unit demand.

We define $R^j(S, i)$ by modifying the definition of $R(S, i)$ in Definition 1.

- $R^j(S, 0) = VCG^j(S)$. $VCG^j(S)$ is the VCG price of item j (the VCG payment from the agent winning item j) when we allocate all the items to the agents in S using VCG.
- For $i = 1, \ldots, |S| - m - 1$, $R^j(S, i)$ equals
$$\frac{1}{m+i}\left(\sum_{a \in S} R^j(S-a, i-1) - (|S| - m - i) R^j(S, i-1)\right).$$

PROPOSITION 3. *For any set of agents S, for $i = 0, \ldots, |S| - m - 1$, we have*

$$\sum_{j=1}^{m} R^j(S, i) = R(S, i)$$

PROOF. We prove by induction. When $i = 0$, by definition, for any S,

$$\sum_{j=1}^{m} R^j(S, 0) = R(S, 0)$$

Now let us assume that for $0 \leq k < |S| - m - 1$,

$$\sum_{j=1}^{m} R^j(S, k) = R(S, k)$$

We have that

$$\sum_{j=1}^{m} R^j(S, k+1)$$

$$= \sum_{j=1}^{m} \frac{1}{m+k+1}(\sum_{a \in S} R^j(S-a, k) - (|S| - m - k - 1) R^j(S, k))$$

$$= \frac{1}{m+k+1}(\sum_{a \in S} R(S-a, k) - (|S| - m - k - 1) R(S, k))$$

$$= R(S, k+1)$$

□

We want to prove that for heterogeneous-item auctions with unit demand, the following redistribution monotonicity condition holds.

$$R(S, 0) \geq R(S, 1) \geq \ldots \geq R(S, |S| - m - 1) \geq 0$$

By Proposition 3, it suffices to prove that for all j,

$$R^j(S, 0) \geq R^j(S, 1) \geq \ldots \geq R^j(S, |S| - m - 1) \geq 0.$$

Without loss of generality, we will prove

$$R^1(S, 0) \geq R^1(S, 1) \geq \ldots \geq R^1(S, |S| - m - 1) \geq 0.$$

To prove the above inequality, we need the following definitions and propositions. *From now on to the end of this section, the setting by default is heterogeneous-item auctions with unit demand, unless specified.*

We use $E(T, S)$ to denote the efficient total valuation when we allocate all the items in T to the agents in S.

PROPOSITION 4. *Submodularity in both items and agents [14]: For any T_1, T_2, S, we have*

$$E(T_1, S) + E(T_2, S) \geq E(T_1 \cup T_2, S) + E(T_1 \cap T_2, S).$$

For any T, S_1, S_2, we have

$$E(T, S_1) + E(T, S_2) \geq E(T, S_1 \cup S_2) + E(T, S_2 \cap S_2).$$

[14] showed that the proposition is true when gross substitutes condition holds. Heterogeneous-item auctions with unit demand satisfies gross substitutes.

We use $\{1\} \oplus \{1, \ldots, m\}$ to denote the item set that contains not only item 1 to m, but also an additional duplicate of item 1.

PROPOSITION 5. *Let S be any set of agents. Let a be the agent who wins item 1 when we allocate the items $\{1, \ldots, m\}$ to the agents in S. We have that $E(\{1\} \oplus \{1, \ldots, m\}, S) = E(\{1\}, a) + E(\{1, \ldots, m\}, S - a)$. That is, after we add an additional duplicate of item 1 to the auction, there exists an efficient allocation under which agent a still wins item 1.*

The above proposition was proved in [11].

PROPOSITION 6. *For any set of agents S, for any $a \in S$, we have $VCG^1(S) \geq VCG^1(S - a)$. That is, the VCG price of item 1 is nondecreasing as the set of agents expands.*

PROOF. Let w_1 be the winner of item 1 when we allocate the items $\{1, \ldots, m\}$ to the agents in S using VCG. $VCG^1(S) = E(\{1, \ldots, m\}, S - w_1) - E(\{2, \ldots, m\}, S - w_1)$. a could be either w_1 or some other agent. We discuss case by case.

Case $a = w_1$: Let w_1' be the new winner of item 1 when we allocate the items $\{1, \ldots, m\}$ to the agents in $S - w_1$ using VCG. $VCG^1(S - w_1) = E(\{1, \ldots, m\}, S - w_1 - w_1') - E(\{2, \ldots, m\}, S - w_1 - w_1')$. We need to prove that $E(\{1, \ldots, m\}, S - w_1) - E(\{2, \ldots, m\}, S - w_1) \geq E(\{1, \ldots, m\}, S - w_1 - w_1') - E(\{2, \ldots, m\}, S - w_1 - w_1')$. We construct a new agent x. Let x's valuation for item 1 be extremely high so that she wins item 1. The above inequality can be rewritten as $E(\{1, \ldots, m\}, S - w_1) - E(\{2, \ldots, m\}, S - w_1) - E(\{1\}, x) \geq E(\{1, \ldots, m\}, S - w_1 - w_1') - E(\{2, \ldots, m\}, S - w_1 - w_1') - E(\{1\}, x)$. This is, $E(\{1,\ldots,m\}, S-w_1) - E(\{1,\ldots,m\}, S-w_1+x) \geq E(\{1,\ldots,m\}, S-w_1-w_1') - E(\{1,\ldots,m\}, S-w_1-w_1'+x)$. We rearrange the terms, and get $E(\{1,\ldots,m\}, S-w_1) + E(\{1,\ldots,m\}, S-w_1-w_1'+x) \geq E(\{1,\ldots,m\}, S-w_1+x) + E(\{1,\ldots,m\}, S-w_1-w_1')$. This inequality can be proved based on Proposition 4.

Case $a \neq w_1$: Let w_1' be the new winner of item 1 when we allocate all the items $\{1,\ldots,m\}$ to the agents in $S-a$ using VCG. $VCG^1(S-a) = E(\{1,\ldots,m\}, S-a-w_1') - E(\{2,\ldots,m\}, S-a-w_1')$. We need to prove that $E(\{1,\ldots,m\}, S-w_1) - E(\{2,\ldots,m\}, S-w_1) \geq E(\{1,\ldots,m\}, S-a-w_1') - E(\{2,\ldots,m\}, S-a-w_1')$. That is, we need to prove $E(\{1,\ldots,m\}, S-w_1) - E(\{2,\ldots,m\}, S-w_1) - E(\{1\}, w_1) - E(\{1\}, w_1') \geq E(\{1,\ldots,m\}, S-a-w_1') - E(\{2,\ldots,m\}, S-a-w_1') - E(\{1\}, w_1) - E(\{1\}, w_1')$. We simplify and rearrange terms, and get $E(\{1,\ldots,m\}, S-w_1) + E(\{1,\ldots,m\}, S-a) + E(\{1\}, w_1) \geq E(\{1,\ldots,m\}, S) + E(\{1,\ldots,m\}, S-a-w_1') + E(\{1\}, w_1')$. Proposition 4 says that $E(\{1,\ldots,m\}, S-a) + E(\{1,\ldots,m\}, S-w_1') \geq E(\{1,\ldots,m\}, S-a-w_1') + E(\{1,\ldots,m\}, S)$. So it suffices to prove $E(\{1,\ldots,m\}, S-w_1) + E(\{1\}, w_1) \geq E(\{1,\ldots,m\}, S-w_1') + E(\{1\}, w_1')$. By Proposition 5, the left-hand side is $E(\{1\} \oplus \{1,\ldots,m\}, S)$. The right-hand side is at most this. □

PROPOSITION 7. *Winners still win after we remove some other agents [4, 6]:*[4] *For any set of agents S and any set of items T, we use W to denote the set of winners when we allocate the items in T to the agents in S using VCG. After we remove some agents in S, those in W that have not been removed remain to be winners,* provided that a consistent tie-breaking rule exists.

It should be noted that there may not exist a consistent tie-breaking rule that satisfies the above proposition. Fortunately, we are able to prove that tie-breaking is irrelevant for the goal of proving redistribution monotonicity.

We say that a type profile is *tie-free* if it satisfies the following: Let $T_1 = \{1\} \oplus \{1,\ldots,m\}$. Let $T_2 = \{1,\ldots,m\}$. Basically, T_1 and T_2 are the only item sets that we will ever mention. A type profile is *tie-free* if for any set of agents S, when we allocate the items in T_1 (or T_2) to S, the set of VCG winners is unique. If we only consider tie-free type profiles, then we do not need to be bothered by tie-breaking. We notice that the set of tie-free type profiles is a *dense* subset of the set of all type profiles – any type profile can be perturbed infinitesimally to become a tie-free type profile.

Our ultimate goal is to prove that for any set of agents S,

$$R(S,0) \geq R(S,1) \geq \ldots \geq R(S,|S|-m-1) \geq 0$$

We notice that the $R(S,j)$ are continuous in the agents' types. Therefore, it suffices to prove the above inequality for tie-free type profiles only.

From now on, we simply assume that the set of VCG winners is always unique.

Definition 3. For any set of agents S with $|S| \geq m+1$, let $D(S)$ be the set of $m+1$ winners when we allocate $\{1\} \oplus \{1,\ldots,m\}$ to the agents in S. $D(S)$ is called the *determination set* of S.

PROPOSITION 8. *For any set of agents S and any $a \in S - D(S)$, we have $VCG^1(S) = VCG^1(S-a)$ and $D(S) = D(S-a)$.*

The above proposition says that for the purpose of calculating item 1's VCG price, only those agents in $D(S)$ are relevant.

[4]The proposition was originally introduced in [4]. A more rigorous proof of a more general claim was also given in [6].

PROOF. $D(S)$ is the set of VCG winners when we allocate $\{1\} \oplus \{1,\ldots,m\}$ to the agents in S. By Proposition 7, after removing $a \in S - D(S)$, every agent in $D(S)$ should still win. That is, $D(S-a) = D(S)$.

Let w_1 be the winner of item 1 when we allocate $\{1,\ldots,m\}$ to the agents in S. $VCG^1(S) = E(\{1,\ldots,m\}, S-w_1) - E(\{2,\ldots,m\}, S-w_1) = E(\{1,\ldots,m\}, S-w_1) + E(\{1\}, w_1) - E(\{1,\ldots,m\}, S) = E(\{1\} \oplus \{1,\ldots,m\}, S) - E(\{1,\ldots,m\}, S)$ (the last step is due to Proposition 5). The first term only depends on those in $D(S)$. The second term also only depends on those in $D(S)$ for the following reason: Let S' be the set of VCG winners when we allocate $\{1,\ldots,m\}$ to the agents in S. The second term only depends on those in S'. We introduce an agent x whose valuation for item 1 is extremely high so that she wins item 1. When we allocate $\{1\} \oplus \{1,\ldots,m\}$ to the agents in $S+x$, the set of VCG winners are then $x + S'$. $D(S)$ are the new set of VCG winners after we remove x. By Proposition 7, those in S' must still remain in $D(S)$. Overall, $VCG^1(S)$ only depends on those agents in $D(S)$. Similarly, $VCG^1(S-a)$ only depends on those agents in $D(S-a)$. For $a \in S - D(S)$, $D(S) = D(S-a)$. Therefore, we must have $VCG^1(S) = VCG^1(S-a)$. □

Definition 4. Let S be any set of agents. Let k be any integer from 1 to $|S|$. Let $a_1 \prec a_2 \prec \ldots \prec a_k$ be a sequence of k distinct agents in S. We say these k agents form a *winner sequence with respect to S* if

$$a_1 \in D(S); a_2 \in D(S-a_1); a_3 \in D(S-a_1-a_2);$$

$$\ldots; a_k \in D(S-a_1-\ldots-a_{k-1}).$$

Let S' be a subset of S of size k. We say that S' forms a winner sequence with respect to S if there exists an ordering of the agents in S' that forms a winner sequence with respect to S. When S' forms a winner sequence with respect to S, we call S' a *size-$|S'|$ winner sequence set* with respect to S.

Let $H(S',S) = 1$ if S' forms a winner sequence with respect to S, and let $H(S',S) = 0$ otherwise. For presentation purpose, we say that the empty set forms a winner sequence (of size 0) with respect to any set S. That is, $H(\emptyset, S) = 1$.

Now we are ready to prove that heterogeneous-item auctions with unit demand satisfies redistribution monotonicity. We recall that it suffices to prove that for any set of agents S,

$$R^1(S,0) \geq R^1(S,1) \geq \ldots \geq R^1(S,|S|-m-1) \geq 0.$$

Here, $R^1(S,0) = VCG^1(S)$, and for $i = 1,\ldots,|S|-m-1$, $R^1(S,i)$ equals

$$\frac{1}{m+i}\left(\sum_{a\in S} R^1(S-a,i-1) - (|S|-m-i)R^1(S,i-1)\right).$$

PROPOSITION 9. *For any set of agents S, $R^1(S,k)$ equals*

$$\frac{1}{\binom{m+k}{m}} \sum_{\substack{S'\subset S\\ |S'|=k\\ H(S',S)=1}} VCG^1(S-S').$$

We have that

$$|\{S' | S' \subset S; |S'| = k; H(S',S) = 1\}| = \binom{m+k}{m}.$$

That is, $R^1(S,k)$ is the average of $VCG^1(S-S')$ for all S' that is a size-k winner sequence set with respect to S. For any set of *agents S (it should be noted that for $R^1(S,k)$ to be well-defined, we need $|S| \geq k+m+1$), the total number of size-k winner sequence sets with respect to S is $\binom{m+k}{m}$.*

The following lemmas are needed for the proof of the above proposition. All these lemmas are implications of "winners still win after we remove some other agents". The proofs are omitted due to space constraints.

LEMMA 1. *Let S be any set of agents. Let S' be a subset of S that forms a winner sequence with respect to S. Let a be an arbitrary agent in $S-S'$. Then, S' must also form a winner sequence with respect to $S-a$.*

LEMMA 2. *Let S be any set of agents. Let a be an agent in S. Let S' be a subset of $S-a$ that forms a winner sequence with respect to $S-a$. If we have that $a \notin D(S-S')$, then S' also forms a winner sequence with respect to S.*

LEMMA 3. *Let S be any set of agents. Let a be an agent in S. Let S' be a subset of $S-a$ that forms a winner sequence with respect to $S-a$. We have that if $a \in D(S-S')$, then $S'+a$ forms a (longer) winner sequence with respect to S.*

LEMMA 4. *Let S be any set of agents. Let $S'+a$ be a subset of S that forms a winner sequence with respect to S. We must have that S' forms a winner sequence with respect to $S-a$ and $a \in D(S-S')$.*

Now we are ready to prove the proposition.

PROOF. We prove by induction.

Initial step: We have $R^1(S,0) = VCG^1(S)$. When $k=0$,

$$\frac{1}{\binom{m}{m}} \sum_{\substack{S' \subset S \\ |S'|=0 \\ H(S',S)=1}} VCG^1(S-S') = VCG^1(S-\emptyset) = VCG^1(S)$$

Also, when $k=0$,

$$|\{S' | S' \subset S; |S'|=0; H(S',S)=1\}| = |\{\emptyset\}| = 1 = \binom{m+0}{m}$$

Induction assumption: We assume that for $k \geq 0$, for any S ($|S| \geq k+m+1$), we have

$$R^1(S,k) = \frac{1}{\binom{m+k}{m}} \sum_{\substack{S' \subset S \\ |S'|=k \\ H(S',S)=1}} VCG^1(S-S')$$

Also, $|\{S' | S' \subset S; |S'|=k; H(S',S)=1\}| = \binom{m+k}{m}$.

We need to prove that the results hold for $k+1$. That is, for any S ($|S| \geq k+m+2$),

$$R^1(S,k+1) = \frac{1}{\binom{m+k+1}{m}} \sum_{\substack{S' \subset S \\ |S'|=k+1 \\ H(S',S)=1}} VCG^1(S-S')$$

and $|\{S' | S' \subset S; |S'|=k+1; H(S',S)=1\}| = \binom{m+k+1}{m}$.

Induction proof: By definition, $R^1(S,k+1)$ equals

$$\frac{1}{m+k+1}\left(\sum_{a \in S} R^1(S-a,k) - (|S|-m-k-1)R^1(S,k)\right) \tag{8}$$

Now let us analyze the expression $\sum_{a \in S} R^1(S-a,k)$. By induction assumption, it can be rewritten as

$$\frac{1}{\binom{m+k}{m}} \sum_{a \in S} \sum_{\substack{S' \subset S-a \\ |S'|=k \\ H(S',S-a)=1}} VCG^1(S-a-S').$$

By induction assumption, the above expression is the sum of $|S|\binom{m+k}{m}$ terms. Each term corresponds to one choice of a among S and one choice of S' among $S-a$. We divide these $|S|\binom{m+k}{m}$ terms into two groups:

Group A, terms with $a \notin D(S-S')$: By Lemma 2, S' must also form a winner sequence with respect to S. That is, there are at most $\binom{m+k}{k}$ choices of S'. For each choice of S', there are at most $|S-S'-D(S-S')| = |S|-k-m-1$ choices of a. Overall, there are at most $\binom{m+k}{k}(|S|-k-m-1)$ terms in Group A. On the other hand, for any S' that forms a winner sequence with respect to S, S' must also form a winner sequence with respect to $S-a$ by Lemma 1. For any $a \notin D(S-S')$, there must be a term in Group A that is characterized by a and S'. That is, there are at least $\binom{m+k}{k}(|S|-k-m-1)$ terms in Group A. Hence, there are exactly $\binom{m+k}{k}(|S|-k-m-1)$ terms in Group A. Since $a \notin D(S-S')$, we have that $VCG^1(S-a-S') = VCG^1(S-S')$ by Proposition 8. Therefore, the sum of all the terms in Group A equals

$$\frac{1}{\binom{m+k}{m}}(|S|-k-m-1) \sum_{\substack{S' \subset S \\ |S'|=k \\ H(S',S)=1}} VCG^1(S-S')$$

This is exactly $|S|-k-m-1$ times $R^1(S,k)$.

Group B, terms with $a \in D(S-S')$: There are exactly $|S|\binom{m+k}{m} - (|S|-k-m-1)\binom{m+k}{m} = (k+m+1)\binom{m+k}{m} = \frac{(k+m+1)!(k+1)}{m!(k+1)!} = (k+1)\binom{m+k+1}{m}$ terms in Group B. Let X be the set of all size-$(k+1)$ winner sequence sets with respect to S. According to Lemma 3 and Lemma 4, every term in Group B must corresponds to an element in X, and every element in X must correspond to exactly $k+1$ terms in Group B (*e.g.*, a size-$(k+1)$ winner sequence set $Y = \{x_1, \ldots, x_{k+1}\}$ corresponds to the following $k+1$ terms: $a = x_i$ and $S' = Y - x_i$ for all i). Therefore, the total number of elements in X must be $\binom{m+k+1}{m}$.

The sum of the terms in Group B equals

$$\frac{k+1}{\binom{m+k}{m}} \sum_{\substack{S' \subset S \\ |S'|=k+1 \\ H(S',S)=1}} VCG^1(S-S')$$

Equation 8 can then be simplified as

$$\frac{1}{m+k+1}\left(\sum_{a \in S} R^1(S-a,k) - (|S|-m-k-1)R^1(S,k)\right)$$

$$= \frac{1}{m+k+1}\left(\frac{k+1}{\binom{m+k}{m}} \sum_{\substack{S' \subset S \\ |S'|=k+1 \\ H(S',S)=1}} VCG^1(S-S')\right)$$

$$= \frac{1}{\binom{m+k+1}{m}} \sum_{\substack{S' \subset S \\ |S'|=k+1 \\ H(S',S)=1}} VCG^1(S - S')$$

□

Proposition 9 implies that function R^1 is always nonnegative. We still need to prove that

$$R^1(S,0) \geq R^1(S,1) \geq \ldots \geq R^1(S,|S|-m-1).$$

Due to space constraint, we only present an outline of the proof of $R^1(S,3) \geq R^1(S,4)$, which highlights the main idea behind the full proof.

PROPOSITION 10. *$R^1(S,3) \geq R^1(S,4)$ for any S. (We need $4 \leq |S|-m-1$ for $R^1(S,4)$ to be well-defined.)*

Proof sketch: By definition, $R^1(S,4) = \frac{1}{m+4}(\sum_{a \in S} R^1(S-a,3) - (|S|-m-4)R^1(S,3))$. To prove that $R^1(S,4) \leq R^1(S,3)$, it suffices to prove that $R^1(S,3) \geq R^1(S-a,3)$ for any $a \in S$.

Let a be an arbitrary agent in S. According to Proposition 9, we need to prove

$$\sum_{\substack{S' \subset S \\ |S'|=3 \\ H(S',S)=1}} VCG^1(S - S') \geq \sum_{\substack{S' \subset S-a \\ |S'|=3 \\ H(S',S-a)=1}} VCG^1(S - a - S').$$

The proof is outlined as follows:

- On both sides of the inequality, there are $\binom{m+3}{m}$ terms (Proposition 9). Every term is characterized by a size-3 winner sequence set S'.
- For every term on the right-hand side, we map it to a corresponding term on the left-hand side. The corresponding term on the left-hand side is larger or the same.
- We prove that the mapping is injective. That is, different terms on the right-hand side are mapped to different terms on the left-hand side.
- Therefore, the left-hand side must be greater than or equal to the right-hand side.

4. CONCLUSION

We conclude our paper with the following conjecture:

CONJECTURE 1. *Gross substitutes implies redistribution monotonicity. That is, HETERO remains feasible and worst-case optimal in combinatorial auctions with gross substitutes.*

The idea is that both multi-unit auctions with nonincreasing marginal values and heterogeneous-item auctions with unit demand satisfy redistribution monotonicity. A natural conjecture is that the "most restrictive joint" of these two settings also satisfies redistribution monotonicity. There are many well-studied auction settings that contain both multi-unit auctions with nonincreasing marginal values and heterogeneous-item auctions with unit demand (a list of which can be found in [10]). Among these well-studied settings, combinatorial auctions with gross substitutes is the most restrictive. To prove the conjecture, we need to prove that gross substitutes implies that for any set of agents S, $R(S,0) \geq R(S,1) \geq \ldots \geq R(S,|S|-m-1) \geq 0$. So far, we have only proved $R(S,0) \geq R(S,1) \geq 0$.

5. REFERENCES

[1] K. Apt, V. Conitzer, M. Guo, and E. Markakis. Welfare undominated Groves mechanisms. In *Proceedings of the Fourth Workshop on Internet and Network Economics (WINE)*, pages 426–437, Shanghai, China, 2008.

[2] M. J. Bailey. The demand revealing process: to distribute the surplus. *Public Choice*, 91:107–126, 1997.

[3] R. Cavallo. Optimal decision-making with minimal waste: Strategyproof redistribution of VCG payments. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 882–889, Hakodate, Japan, 2006.

[4] S. Gujar and Y. Narahari. Redistribution mechanisms for assignment of heterogeneous objects. *J. Artif. Intell. Res. (JAIR)*, 41:131–154, 2011.

[5] F. Gul and E. Stacchetti. Walrasian equilibrium with gross substitutes. *Journal of Economic Theory*, 87:95–124, 1999.

[6] M. Guo. VCG redistribution with gross substitutes. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, San Francisco, CA, USA, 2011.

[7] M. Guo and V. Conitzer. Worst-case optimal redistribution of VCG payments in multi-unit auctions. *Games and Economic Behavior*, 67(1):69–98, 2009.

[8] B. Holmström. Groves' scheme on restricted domains. *Econometrica*, 47(5):1137–1144, 1979.

[9] A. Kelso and V. Crawford. Job matching, coalition formation, and gross substitues. *Econometrica*, 50:1483–1504, 1982.

[10] B. Lehmann, D. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 2003.

[11] H. Leonard. Elicitation of honest preferences for the assignment of individual to positions. *Journal of Political Economics*, 91(3):461–479, 1983.

[12] H. Moulin. Almost budget-balanced VCG mechanisms to assign multiple objects. *Journal of Economic Theory*, 144(1):96–119, 2009.

[13] V. Naroditskiy, M. Polukarov, and N. R. Jennings. Optimization in payments in dominant strategy mechanisms for single-parameter domains. In *Proceedings of the Third International Workshop on Computational Social Choice (COMSOC)*, Dusseldorf, Germany, 2010.

[14] M. Yokoo, Y. Sakurai, and S. Matsubara. The effect of false-name bids in combinatorial auctions: New fraud in Internet auctions. *Games and Economic Behavior*, 46(1):174–188, 2004.

APPENDIX

PROPOSITION 11. *Let M be a feasible VCG redistribution mechanism that is possibly not anonymous. Let α^M be the worst-case redistribution fraction of M. If the agents' type spaces are identical, then there exists an anonymous feasible VCG redistribution mechanism, whose worst-case redistribution fraction is at least α^M.*

Proof sketch: M can be anonymized using the technique described in Section 3 of [1]. The resulting mechanism is anonymous, feasible, and its worst-case redistribution fraction is at least α^M.

False-name-proofness in Online Mechanisms

Taiki Todo, Takayuki Mouri, Atsushi Iwasaki, and Makoto Yokoo
Department of Informatics, Kyushu University
Motooka 744, Fukuoka, Japan
{todo@agent., mouri@agent., iwasaki@, yokoo@}inf.kyushu-u.ac.jp

ABSTRACT

In real electronic markets, each bidder arrives and departs over time. Thus, such a mechanism that must make decisions dynamically without knowledge of the future is called an *online mechanism.* In an online mechanism, it is very unlikely that the mechanism designer knows the number of bidders beforehand or can verify the identity of all of them. Thus, a bidder can easily submit multiple bids (*false-name bids*) using different identifiers (e.g., different e-mail addresses). In this paper, we formalize false-name manipulations in online mechanisms and identify a simple property called (value, time, identifier)-monotonicity that characterizes the allocation rules of false-name-proof online auction mechanisms. To the best of our knowledge, this is the first work on false-name-proof online mechanisms. Furthermore, we develop a new false-name-proof online auction mechanism for k identical items. When $k = 1$, this mechanism corresponds to the optimal stopping rule of the secretary problem where the number of candidates is unknown. We show that the competitive ratio of this mechanism for efficiency is 4 and independent from k by assuming that only the distribution of bidders' arrival times is known and that the bidders are impatient.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multi-agent systems*; J.4 [**Social and Behavioral Sciences**]: Economics

General Terms

Algorithms, Economics, Theory

Keywords

Auctions, mechanism design, game theory, online algorithms

1. INTRODUCTION

Auctions have become an integral part of electronic commerce and a promising application field of game theory and mechanism design theory. Traditionally, mechanism design of auctions has mainly considered static (offline) environments where all bidders arrive and depart simultaneously and the mechanism makes a decision only one time. In real electronic markets, however, each bidder may arrive and depart over time, so a mechanism must make decisions dynamically without knowledge of the future. This uncertainty often makes traditional works inapplicable to such online environments. Therefore, designing mechanisms for dynamic environments (i.e., *online mechanism design*) has lately attracted considerable attention in the algorithmic game-theory field [14].

One desirable characteristic of a mechanism is *strategy-proofness.* A mechanism is strategy-proof if for each bidder, truthfully reporting her type (private information) is a dominant strategy. Unlike traditional mechanism design environments, in online mechanisms, the private information of each bidder consists not only of his valuation but also of her arrival/departure times, and bidders can misreport them to maximize their utility. For these reasons, designing a strategy-proof mechanism is much more challenging in online environments than in traditional static environments.

Designing a strategy-proof online mechanism is strongly connected to the optimal stopping theory, in particular, the secretary problem. In fact, from the perspective of this theory, Hajiaghayi et al. [9] developed an online mechanism for a single item where an auction with a single item is held in finite periods. Hereafter, we refer to it as Mechanism 1. Consider there are n bidders. Each bidder $i \in N$ values the item at r_i and stays in the auction at interval $[a_i, d_i]$.

MECHANISM 1. *Let n denote the number of bidders and a^* be the arrival time of $\lfloor n/e \rfloor$-th bidder, where e is the base of the natural logarithm.*

1. *(learning phase): At period a^*, let $r_{(1)}, r_{(2)}$ be the first and second highest bidding values received so far.*
2. *(transition): If a bidder whose bidding value is $r_{(1)}$ remains present at period a^*, then sell an item to that bidder (breaking ties deterministically, e.g., based on the lexicographic order of the identifiers) at price $r_{(2)}$.*
3. *(accepting phase): Otherwise, sell an item to the next bidder whose bidding value is at least $r_{(1)}$ (breaking ties deterministically) at price $r_{(1)}$.*

Mechanism 1 is strategy-proof assuming no bidder can be present longer than her true stay. A winner cannot decrease her payment by making her stay shorter or by misreporting her valuation. A loser cannot win unless she pays more than her true valuation or stays longer.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

Table 1: False-name-proofness fails in Mechanism 1. a_i, d_i, and r_i indicate arrival period, departure period, and the valuation of bidder i.

	a_i	d_i	r_i		a_i	d_i	r_i
bidder 1	1	3	6	bidder 4	(later)		7
bidder 2	4	5	2	bidder 5	(later)		4
bidder 3	4	4	8	bidder 6	(later)		1

However, untruthfully declaring private information is only one way to manipulate the outcome. Another way is for one bidder to pretend to be multiple bidders. Such *false-name bids* [16], i.e., bids submitted under fictitious names such as multiple e-mail addresses, are especially feasible in Internet auctions due to their relative anonymity. Unfortunately, Mechanism 1 is not false-name-proof; a bidder can profit by pretending to be multiple bidders.

EXAMPLE 1. *Consider a single item online auction with six bidders, each of whom has a preference, as shown in Table 1. Since $n = 6$, the mechanism waits for the second bidder ($\lfloor 6/e \rfloor = 2$). When each bidder reports truthfully, the item is not sold until arrival period 4 of bidder 2. In this case, bidder 1 cannot win even if her bidding value is very high, because she is not present when the winner is determined. Next, consider the case when bidder 1 uses two identifiers, $1'$ and $1''$. Identifier $1'$ keeps her bid, and identifier $1''$ reports $(2, 2, \epsilon)$. In this case, the transition to the accepting phase occurs after $\lfloor 7/e \rfloor = 2$ bids. Bidder 1 wins at period 2 and pays ϵ.*

The example shows that false-name bids are profitable for bidder 1. In fact, bidder 1 can also win if she once departs from the auction at period 2 and arrives again at period 3 using another identifier $1''$.

Furthermore, in such environments where bidders can use multiple identifiers, the number of participating bidders (identifiers) n depends on the strategies of the bidders. This means that a mechanism cannot observe correct information about the number of participating bidders; it can only observe the number of identifiers used by the bidders. Thus, it is impractical to design an online mechanism that is based on the fact that the mechanism knows the number of participating identifiers n in advance. This difficulty was also pointed out by [10].

Readers might think that if a market can use some personal identification method (e.g., checking the participant's credit card number or social security number), the problem resulting from false-name bids disappears. Introducing such a method can indeed slightly increase the cost of using false-name bids, but it cannot completely solve the problem. A person can ask his/her family, friends, or employers to submit bids on her/his behalf. False-name manipulations can be considered as a very restricted subclass of collusions, where a person can only collude with other participants when they were initially not interested in participating in the mechanism, but they agree to work on behalf of the person by obtaining a small side-payment. Such manipulations cannot be prevented by a simple personal identification method. Conitzer and Yokoo [4] provided a more detailed discussion why false-name-proof mechanisms matter.

Our Results.

To the best of our knowledge, this is the first work that deals with false-name manipulations in online mechanisms. This paper formalizes false-name manipulations in online mechanisms and proposes a simple property called *(value, time, identifier)-monotonicity*, which characterizes false-name-proof online auction mechanisms in single-valued domains. Then it introduces two non-trivial false-name-proof mechanisms for k identical items. Furthermore, the competitive analysis revealed that for sufficiently large k, one of them is 4-competitive for efficiency by introducing a different adversarial model from the traditional one under the assumption that all bidders are impatient. We assume here that a mechanism has no information about the number of bidders; it does know the distribution of their arrival times, since it is quite natural that their real number is unknown and unpredictable in situations where false-name bids are possible.

Related Work.

Lavi and Nisan [12] was the first work on mechanism design of auctions in dynamic environments. Hajiaghayi et al. [9] proposed a strategy-proof online mechanism in limited-supply environments, based on the optimal stopping rule of the secretary problem. Hajiaghayi et al. [8] proposed a strategy-proof online mechanism for selling expiring items. In Hajiaghayi et al. [10], a technique called automated mechanism design was applied to construct online auction mechanisms. Furthermore, Parkes [14] showed that the revelation principle can fail in online mechanisms when the no-early arrival, no-late departure property does not hold. Gerding et al. [5] introduced two procedures for item burning into online mechanisms to achieve truthfulness.

Yokoo et al. [16] pointed out the effects of false-name manipulations in combinatorial auctions and showed that even the Vickrey-Clarke-Groves (VCG) mechanism is vulnerable against false-name manipulations. Besides combinatorial auctions, the notion of false-name-proofness have been discussed in other application fields of game theory, such as resource allocation [7] and coalitional games [1].

Myerson [13] proposed the *monotonicity* property of allocation rules, which characterizes strategy-proof auction mechanisms in single-parameter settings. Bikhchandani et al. [2] extended the property to such multi-dimensional settings as combinatorial auctions and proposed a property called *weak-monotonicity* that characterizes strategy-proof mechanisms. Todo et al. [15] proposed the *sub-additivity* property as a full characterization of false-name-proof combinatorial auction mechanisms. For online auction mechanisms, Hajiaghayi et al. [8] and Parkes [14] introduced a property called *monotonicity* [1] and showed that it characterizes strategy-proof online auction mechanisms.

2. PRELIMINARIES

Let $N = \{1, 2, \dots n\}$ denote a set of bidders and $\mathbb{T} = \{1, \dots, T\}$ a set of finite and discrete time periods in which an auction is held. Each bidder $i \in N$ has private information, or a type, $\theta_i = (a_i, d_i, r_i)$ drawn from Θ_i. The type of bidder defines its value for the allocations of an online mechanism. Θ_i is a type space, or a domain of types, defined as

[1]To distinguish this property from the original monotonicity introduced by Myerson [13], we refer to it as *(value, time)-monotonicity.*

$\Theta_i = \mathbb{T} \times \mathbb{T} \times \mathbb{R}_{\geq 0}$. Let a_i and d_i be *arrival* and *departure* times. In the interval of a_i and d_i, a bidder has a valuation r_i on the auctioned item. Define $x = (x^1, \ldots, x^T) \in X$ as a possible allocation in a mechanism. Each $x^t = (x_1^t, \ldots, x_n^t)$ represents the allocation at period $t \in \mathbb{T}$, where x_i^t is the allocation to bidder i at period t; if bidder i is allocated an item at period t, then $x_i^t = 1$ holds; otherwise $x_i^t = 0$. We represent the gross utility of bidder i whose type is θ_i for an allocation x as $v(\theta_i, x)$.

We restrict the domain of types Θ_i to *single-valued* domains [14], in which each $\theta_i \in \Theta_i$ is defined as a triple (a_i, d_i, r_i), where the gross utility of bidder i whose type is θ_i is defined as follows:

$$v(\theta_i, x) = \begin{cases} r_i & \text{if } x_i^t = 1 \text{ holds for some } t \in [a_i, d_i] \\ 0 & \text{otherwise.} \end{cases}$$

We also assume a *quasi-linear* utility; the net utility of bidder i who obtains at least one item during her stay and pays p is represented as $v(\theta_i, x) - p = r_i - p$.

An online mechanism $\mathcal{M}(f, p)$ consists of an allocation rule f and a payment rule p. An allocation rule f is defined as $f = \{f^t | t \in \mathbb{T}\}$. Here, $\theta = (\theta_1, \ldots, \theta_n)$ denotes a type profile reported by a set of bidders N, and $\Theta = \times_{i \in N} \Theta_i$ denotes a set of possible type profiles. Each $f^t : \Theta \to \{0, 1\}^n$ is a mapping from a set of reported type profiles to a set of possible allocations. Let $f_i(\theta_i, \theta_{-i})$ denote the allocation to bidder i where θ_i is the declared type of bidder i and θ_{-i} is the declared type profile of other bidders. A payment rule p is defined as $p = (p_1, \ldots, p_n)$. Each $p_i : \Theta \to \mathbb{R}_{\geq 0}$ is a mapping from a set of type profiles to a set of non-negative real numbers. Notice that bidder i's reported type $\theta_i' = (a_i'.d_i', r_i')$ is not necessarily the same as her true type $\theta_i = (a_i, d_i, r_i)$. However, we assume no bidder can be present longer than her true stay, i.e., the *no-early arrival, no-late departure* property holds; a reported type θ_i' satisfies $a_i' \geq a_i$ and $d_i' \leq d_i$.

In this paper, we restrict our attention to *direct-revelation, deterministic* online mechanisms. Also, we assume that a mechanism is *almost anonymous* and *individually rational*. A mechanism is almost anonymous if the obtained results are invariant under the permutation of identifiers, except for ties where several bidders have an identical type but their allocations are different (e.g., only one winner). We assume the net utilities of bidders involved in tie-breaking must be the same. Individual rationality means that no participant suffers any loss in a dominant strategy equilibrium; i.e., the payment never exceeds the gross utility of the allocated items. Thus, a mechanism does not collect any payment from losers.

Now, let us define *strategy-proofness*.

Definition 1 (Strategy-proofness). *An online mechanism $\mathcal{M}(f, p)$ is strategy-proof if $\forall i, \theta_{-i}, \theta_i, \theta_i'$,*

$$v(\theta_i, f_i(\theta_i, \theta_{-i})) - p_i(\theta_i, \theta_{-i}) \geq v(\theta_i, f_i(\theta_i', \theta_{-i})) - p_i(\theta_i', \theta_{-i}).$$

A strategy-proof allocation rule is fully characterized by a simple property called *(value, time)-monotonicity* in the single-valued domain [8]. To define the monotonicity property, let us first introduce a concept of *critical value*, which plays an important role for guaranteeing strategy-proofness. In words, a critical value cv is the minimal (threshold) value for a bidder to be a winner.

$$\mathrm{cv}(a_i, d_i, \theta_{-i}) = \begin{cases} \inf r_i \text{ s.t. } f_i((a_i, d_i, r_i), \theta_{-i}) = 1 \\ \infty, \text{ if no such } r_i \text{ exists.} \end{cases} \quad (1)$$

Definition 2 ((value, time)-monotonicity). *An allocation rule f is (value, time)-monotonic if $\forall i, \theta_{-i}, \theta_i = (a_i, d_i, r_i)$, $\theta_i' = (a_i', d_i', r_i')$, the following condition holds:*

$$\begin{aligned} &\text{if } f_i(\theta_i', \theta_{-i}) = 1 \wedge\ r_i' > \mathrm{cv}(a_i', d_i', \theta_{-i}) \\ &\wedge\ a_i \leq a_i' \leq d_i' \leq d_i \wedge\ r_i \geq r_i' \\ &\text{then } \ f_i(\theta_i, \theta_{-i}) = 1. \end{aligned}$$

Note that the condition $r_i' > \mathrm{cv}(a_i', d_i', \theta_{-i})$ is necessary to prevent inconsistent allocations due to tie-breaking, e.g., bidder i and j have the same type. If valuation r_i is strictly greater than r_i', we do not need this condition.

Hajiaghayi et al. [8] proved that if and only if an allocation rule is (value, time)-monotonic, we can find an appropriate payment rule that truthfully implements it in a dominant strategy equilibrium. In addition, it is straightforward to derive such an appropriate payment rule so that an online mechanism $\mathcal{M}(f, p)$ is strategy-proof:

$$p_i(\theta_i, \theta_{-i}) = \begin{cases} \mathrm{cv}(a_i, d_i, \theta_{-i}), & \text{if } f_i(\theta_i, \theta_{-i}) = 1 \\ 0, & \text{otherwise.} \end{cases}$$

When bidder i reports a shorter stay, her payment does not decrease, since bidder i's critical value does not decrease if an allocation rule is (value, time)-monotonic.

In this paper, we focus on a worst-case analysis (*competitive analysis*) to consider the performance of mechanisms. Such analysis is commonly used in recent mechanism design literature, especially by computer scientists. Let us define the competitive ratios for efficiency and revenue. Here, $z \in Z$ denotes the set of inputs available to the adversary and θ^z the corresponding type profile.

Definition 3 (Competitive Ratio for Efficiency). *An online mechanism $\mathcal{M}(f, p)$ is c-competitive for efficiency if for some constant c,*

$$\min_{z \in \mathcal{Z}} \mathbb{E}\{\mathrm{Val}(f(\theta^z))/V^*(\theta^z)\} \geq 1/c.$$

For efficiency, $\mathrm{Val}(f(\theta^z))$ indicates the social surplus of the decision made by an allocation rule f given input θ^z. $V^*(\theta^z)$ indicates the surplus of the best possible allocation obtained by an offline mechanism. This expectation is taken with respect to the random choice derived from the model of an adversary.

Definition 4 (Competitive Ratio for Revenue). *An online mechanism $\mathcal{M}(X, p)$ is c-competitive for revenue if for some constant c,*

$$\min_{z \in \mathcal{Z}} \mathbb{E}\{\mathrm{Rev}(p(\theta^z))/R^*(\theta^z)\} \geq 1/c.$$

For revenue, $R^*(\theta^z)$ indicates the revenue achieved by $\mathcal{F}^{(2,k)}$ auction with k items [6]. The revenue as a benchmark is used in [8]. Here, if $k \geq 2$, $R^*(\theta^z) = \max_{2 \leq m \leq k} m \cdot r_{(m)}$, where $r_{(m)}$ denotes the m-th highest value among all bidders. If $k = 1$, we use VCG revenue of $r_{(2)}$ as the benchmark.

Next, we introduce several notations for discussing false-name-proofness in online auction mechanisms. Let ϕ_i denote the set of identifiers owned by bidder i. Let $\mathbb{N}$ denote a set of identifiers, i.e., $\mathbb{N} = \bigcup_{i \in N} \phi_i$, where N denotes a set of real bidders. Let us re-define θ as the type profile reported by all identifiers. Here, $\mathbf{0}$ indicates that the identifier is not used by its owner. Furthermore, let θ_{ϕ_i} denote a type profile reported by a set of identifiers ϕ_i and $\theta_{-\phi_i}$ a type profile reported by identifiers except for ϕ_i. Using these notations, the allocation to an identifier j when the set of identifiers ϕ_i reports θ_{ϕ_i} and the other identifiers reports $\theta_{-\phi_i}$ is represented as $f_j(\theta_{\phi_i}, \theta_{-\phi_i})$.

Definition 5 (False-name-proofness). *An online mechanism $\mathcal{M}(f,p)$ is false-name-proof if $\forall i$, ϕ_i, $\theta_{-\phi_i}$, θ_i, θ_{ϕ_i}, the following inequality holds:*

$$\begin{aligned}&v(\theta_i, f_i((\theta_i, \mathbf{0}, \ldots, \mathbf{0}), \theta_{-\phi_i})) - p_i((\theta_i, \mathbf{0}, \ldots, \mathbf{0}), \theta_{-\phi_i})\\ &\geq v(\theta_i, \textstyle\sum_{j\in\phi_i} f_j(\theta_{\phi_i}, \theta_{-\phi_i})) - \textstyle\sum_{j\in\phi_i} p_j(\theta_{\phi_i}, \theta_{-\phi_i})\end{aligned}$$

A mechanism is *false-name-proof* if it is a dominant strategy for each bidder to report her true type using a single identifier (although the bidder can use multiple identifiers). When $|\phi_i| = 1$, this definition is identical to Definition 1.

3. CHARACTERIZATION OF FALSE-NAME-PROOFNESS

In this section, we propose a simple property called *(value, time, identifier)-monotonicity* that characterizes false-name-proof allocation rules in online auction mechanisms.

Definition 6 ((value, time, identifier)-monotonicity). *An allocation rule f is (value, time, identifier)-monotonic if for any i, ϕ_i, θ_i, $\theta_{-\phi_i}$, θ_{ϕ_i}, the following holds:*

$$\begin{aligned}&\textit{if } \exists j' \in \phi_i \textit{ s.t.,}\\ &\quad \big(f_{j'}(\theta_{\phi_i}, \theta_{-\phi_i}) = 1 \wedge\ r_{j'} > \mathrm{cv}(a_{j'}, d_{j'}, \theta_{\phi_i\setminus\{j'\}} \cup \theta_{-\phi_i})\big)\\ &\wedge \big(\forall j \in \phi_i, a_i \leq a_j \leq d_j \leq d_i\big)\ \wedge\ r_i \geq \textstyle\sum_{j'\in\phi_i : j' \textit{ wins}} r_{j'}\\ &\text{then}\ \ f_i((\theta_i, \mathbf{0}, \ldots, \mathbf{0}), \theta_{-\phi_i}) = 1.\end{aligned} \tag{2}$$

Note that $\theta_{\phi_i\setminus\{j'\}}$ denote the type profile by the set of identifiers $\phi_i \setminus \{j'\}$. Thus, $\mathrm{cv}(a_{j'}, d_{j'}, \theta_{\phi_i\setminus\{j'\}} \cup \theta_{-\phi_i})$ indicates the critical value of a bidder that stays $[a_{j'}, d_{j'}]$ when the other identifiers reports $\theta_{\phi_i\setminus\{j'\}} \cup \theta_{-\phi_i}$.

Now let us provide an illustrative example of an allocation rule that satisfies Def. 6. Assume that the set of identifiers ϕ_i surrounded by the dashed rectangle in Fig. 1 (a) is owned by bidder i and that the identifier j' with value $r_{j'}$ wins an item. If θ_{ϕ_i} in Fig. 1 (a) is replaced by one type $\theta_i = (a_i, d_i, r_{j'} + \epsilon)$ in Fig. 1 (b), then the allocation rule that satisfies (value, time, identifier)-monotonicity must choose θ_i as a winner, as long as θ_i satisfies the following conditions: (i) the interval $[a_i, d_i]$ includes $[a_j, d_j]$ for all $j \in \phi_i$, (ii) the value $r_{j'} + \epsilon$ exceeds the winner's value $r_{j'}$.

Intuitively, in the inequality $r_i \geq \sum_{j'\in\phi_i : j' \text{ wins}} r_{j'}$ in Def. 6, each term $r_{j'}$ on the right-hand side corresponds to a payment of each winning identifier j' that may be owned by i. To avoid false-name manipulations, bidder i with value r_i that exceeds the sum of $r_{j'}$ must win an item. Otherwise, bidder i has an incentive to manipulate using the set of identifiers ϕ_i. We remark that this property is inspired by two characterizations: (value, time)-monotonicity by Hajiaghayi et al. and sub-additivity by Todo et al. In fact, the property becomes equivalent to (value, time)-monotonicity when $|\phi_i| = 1$ for all i and to sub-additivity when $T = 1$, i.e., in offline auction settings.

Before showing our characterization theorem, let us provide the following lemma:

Lemma 1. *Given an allocation rule that satisfies (value, time, identifier)-monotonicity, the critical value of bidder i is independent of her valuation r_i and weakly increasing in shorter stay and more identifiers.*

Proof. In Parkes [14], it was shown that the critical value is weakly increasing in shorter stay. Then we can show that the critical value is weakly increasing in a larger number of rivals. Now assume that there exists an additional identifier $\theta_{i'} = (a_{i'}, d_{i'}, r_{i'})$ such that $a_i \leq a_{i'} \leq d_{i'} \leq d_i$ and $\mathrm{cv}(a_i, d_i, \theta_{-i}) > \mathrm{cv}(a_i, d_i, \theta_{-i} \cup \theta_{i'})$ to derive a contradiction.

Here, modify the valuation of type θ_i such that $r_i = \mathrm{cv}(a_i, d_i, \theta_{-i} \cup \theta_{i'})$. Then, from the definition of cv when the other bidders (identifiers) report $\theta_{-i} \cup \theta_{i'}$, $f_i(\theta_i, \theta_{-i} \cup \theta_{i'}) = 1$. Also, from the definition of cv when the other bidders (identifiers) report θ_{-i}, $f_i(\theta_i, \theta_{-i}) = 0$. Thus, by setting $\theta_{\phi_i} = \{\theta_i, \theta_{i'}\}$, $\theta_{-\phi_i} = \theta_{-i}$ and $\theta_{j'} = \theta_i$, we have

$$\begin{aligned}&\exists j' \in \phi_i \text{ s.t.,}\\ &\quad \big(f_{j'}(\theta_{\phi_i}, \theta_{-\phi_i}) = 1 \wedge\ r_{j'} > \mathrm{cv}(a_{j'}, d_{j'}, \theta_{\phi_i\setminus\{j'\}} \cup \theta_{-\phi_i})\big)\\ &\wedge \big(\forall j \in \phi_i, a_i \leq a_j \leq d_j \leq d_i\big) \wedge\ r_i \geq \textstyle\sum_{j'\in\phi_i : j' \text{ wins}} r_{j'}\\ &\text{and } f_i((\theta_i, \mathbf{0}, \ldots, \mathbf{0}), \theta_{-\phi_i}) = 0,\end{aligned}$$

which violates (value, time, identifier)-monotonicity. □

Theorem 1. *On a single-valued domain, there always exists an appropriate payment rule p so that an online mechanism $\mathcal{M}(f,p)$ is false-name-proof if and only if the allocation rule f satisfies (value, time, identifier)-monotonicity.*

Proof. **(only if part)** We first prove that if an online mechanism $\mathcal{M}(f,p)$ is false-name-proof, then the allocation rule f satisfies (value, time, identifier)-monotonicity. Parkes [14] proved that if $\mathcal{M}$ is strategy-proof, then f satisfies (value, time)-monotonicity. Since the definition of false-name-proofness is a generalization of strategy-proofness, if $\mathcal{M}$ is false-name-proof, then it is also strategy-proof. Thus, we can assume that f satisfies (value, time)-monotonicity and that p is determined based on the *critical* values in Eq. 1.

We derive a contradiction by assuming that the allocation rule f does not satisfy (value, time, identifier)-monotonicity. More specifically, we assume for bidder i (with type θ_i), who owns the set of identifiers ϕ_i, the following condition holds:

$$\begin{aligned}&\exists j' \in \phi_i \text{ s.t.,}\\ &\quad \big(f_{j'}(\theta_{\phi_i}, \theta_{-\phi_i}) = 1 \wedge\ r_{j'} > \mathrm{cv}(a_{j'}, d_{j'}, \theta_{\phi_i\setminus\{j'\}} \cup \theta_{-\phi_i})\big)\\ &\wedge \big(\forall j \in \phi_i, a_i \leq a_j \leq d_j \leq d_i\big) \wedge\ r_i \geq \textstyle\sum_{j'\in\phi_i : j' \text{ wins}} r_{j'}\\ &\text{and } f_i((\theta_i, \mathbf{0}, \ldots, \mathbf{0}), \theta_{-\phi_i}) = 0.\end{aligned} \tag{3}$$

When Eq. 3 holds, bidder i with type θ_i cannot win the item by truthfully reporting her type. Thus,

$$v(\theta_i, f_i((\theta_i, \mathbf{0}, \ldots, \mathbf{0}), \theta_{-\phi_i})) - p_i((\theta_i, \mathbf{0}, \ldots, \mathbf{0}), \theta_{-\phi_i}) = 0.$$

Also, Eq. 3 implies that if bidder i reports θ_{ϕ_i} using false identifiers, she wins at least one item. Note that since there exists at least one winning identifier j' in ϕ_i such that $r_{j'} > \mathrm{cv}(a_{j'}, d_{j'}, \theta_{\phi_i\setminus\{j'\}} \cup \theta_{-\phi_i})$, we have $\sum_{j'\in\phi_i : j' \text{ wins}} r_{j'} > \sum_{j'\in\phi_i : j' \text{ wins}} p_{j'}(\theta_{\phi_i}, \theta_{-\phi_i})$. Thus,

$$\begin{aligned}&v(\theta_i, \textstyle\sum_{j\in\phi_i} f_j(\theta_{\phi_i}, \theta_{-\phi_i})) - \textstyle\sum_{j\in\phi_i} p_j(\theta_{\phi_i}, \theta_{-\phi_i})\\ &> r_i - \textstyle\sum_{j'\in\phi_i : j' \text{ wins}} r_{j'} \geq 0.\end{aligned}$$

This bidder can increase her utility by using false identifiers, contradicting the assumption of false-name-proofness.

(if part) Next we prove that if an allocation rule f satisfies (value, time, identifier)-monotonicity, then there exists an appropriate payment rule p such that $\mathcal{M}(f,p)$ is false-name-proof. We derive a contradiction by assuming that

$$\begin{aligned}&\forall p, \exists \theta_i, \theta_{\phi_i},\\ &v(\theta_i, f_i((\theta_i, \mathbf{0}, \ldots, \mathbf{0}), \theta_{-\phi_i})) - p_i((\theta_i, \mathbf{0}, \ldots, \mathbf{0}), \theta_{-\phi_i})\\ &< v(\theta_i, \textstyle\sum_{j\in\phi_i} f_j(\theta_{\phi_i}, \theta_{-\phi_i})) - \textstyle\sum_{j\in\phi_i} p_j(\theta_{\phi_i}, \theta_{-\phi_i})\end{aligned} \tag{4}$$

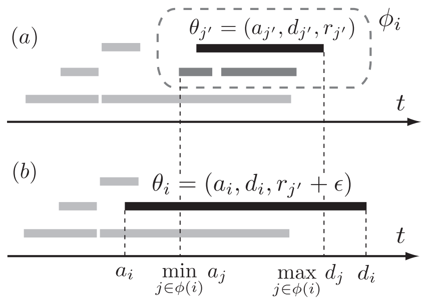

(a)
$\theta_{j'} = (a_{j'}, d_{j'}, r_{j'})$
ϕ_i
t
(b)
$\theta_i = (a_i, d_i, r_{j'} + \epsilon)$
t
a_i
$\min_{j\in\phi(i)} a_j$
$\max_{j\in\phi(i)} d_j$
d_i

Figure 1: Example of allocation rule that satisfies (value, time, identifier)-monotonicity

holds. More specifically, we show that if a payment rule p is defined by a critical value (as Eq. 1), Eq. 4 does not hold in the following two cases: (I) bidder i is winning when she reports truthfully, and (II) bidder i is losing.

Case I: $f_i((\theta_i, \mathbf{0}, \ldots, \mathbf{0}), \theta_{-\phi_i}) = 1$ holds. Since we assume a single-valued domain, the two terms $v(\theta_i, \cdot)$ in Eq. 4 are equivalent. Thus, from Eq. 4, $p_i((\theta_i, \mathbf{0}, \ldots, \mathbf{0}), \theta_{-\phi_i}) > \sum_{j\in\phi_i} p_j(\theta_{\phi_i}, \theta_{-\phi_i})$. Furthermore, since we assume the mechanism does not collect payments from losers, we obtain

$$p_i((\theta_i, \mathbf{0}, \ldots, \mathbf{0}), \theta_{-\phi_i}) > \sum_{j'\in\phi_i : j' \text{ wins}} p_{j'}(\theta_{\phi_i}, \theta_{-\phi_i}). \quad (5)$$

On the other hand, from Lemma 1, the critical value of bidder i with stay $[a_i, d_i]$ weakly increases in shorter stay and more identifiers. Thus, for all winners $j' \in \phi_i$, $\text{cv}(a_i, d_i, \theta_{-\phi_i}) \le \text{cv}(a_{j'}, d_{j'}, \theta_{\phi_i \setminus \{j'\}} \cup \theta_{-\phi_i})$, and therefore

$$\text{cv}(a_i, d_i, \theta_{-\phi_i}) \le \sum_{j'\in\phi_i : j' \text{ wins}} \text{cv}(a_{j'}, d_{j'}, \theta_{\phi_i \setminus \{j'\}} \cup \theta_{-\phi_i}). \quad (6)$$

Thus, with a payment rule p defined by Eq. 1, we obtain

$$p_i((\theta_i, \mathbf{0}, \ldots, \mathbf{0}), \theta_{-\phi_i}) \le \sum_{j'\in\phi_i : j' \text{ wins}} p_{j'}(\theta_{\phi_i}, \theta_{-\phi_i}), \quad (7)$$

and this contradicts Eq. 5.

Case II: $f_i((\theta_i, \mathbf{0}, \ldots, \mathbf{0}), \theta_{-\phi_i}) = 0$ holds. Assume that bidder i, whose true type is $\theta_i = (a_i, d_i, r_i)$, cannot win when she reports truthfully. That is, $r_i < \text{cv}(a_i, d_i, \theta_{-\phi_i})$, and when she reports truthfully her utility is zero.

Here, applying the same argument as in Eq. 6, we have

$$r_i < \sum_{j'\in\phi_i : j' \text{ wins}} \text{cv}(a_{j'}, d_{j'}, \theta_{\phi_i \setminus \{j'\}} \cup \theta_{-\phi_i}).$$

The right-hand side corresponds to the total payment of bidder i when she uses false identifiers θ_{ϕ_i}. This implies that her utility with this manipulation is negative. Thus,

$$v(\theta_i, f_i((\theta_i, \mathbf{0}, \ldots, \mathbf{0}), \theta_{-\phi_i})) - p_i((\theta_i, \mathbf{0}, \ldots, \mathbf{0}), \theta_{-\phi_i}) \\ = 0 > v(\theta_i, \textstyle\sum_{j\in\phi_i} f_j(\theta_{\phi_i}, \theta_{-\phi_i})) - \sum_{j\in\phi_i} p_j(\theta_{\phi_i}, \theta_{-\phi_i})$$

holds, which contradicts Eq. 5. □

To show that the allocation rule of Mechanism 1 does not satisfy (value, time, identifier)-monotonicity, consider Example 1 again. When bidder 1 uses two identifiers, $1'$ and $1''$, seven identifiers participate in the auction ($n = 7$). Since Mechanism 1 waits for the second bidder, identifier $1'$ with $(1, 3, 6)$ obtains the item using identifier $1''$ with $(2, 2, \epsilon)$. However, when bidder 1 uses only one identifier, six identifiers participate in the auction ($n = 6$). In this case, the situation becomes identical to Table 1 and bidder 1 no longer obtains the item. Thus, this allocation rule does not satisfy (value, time, identifier)-monotonicity.

Note that our characterization is constructed on a single-valued domain and thus can be applied to any environment on the domain. For example, (value, time, identifier)-monotonicity is applicable to *expiring-item* environments, where a mechanism allocates a single indivisible item to a bidder in each period, e.g., the right to use a shared computer or a network resource. We then show a representative strategy-proof mechanism for expiring-item environments, called *greedy auction*, and verify whether the allocation rule satisfies (value, time, identifier)-monotonicity.

Mechanism 2 (Greedy Auction [14]). *In each period $t \in \mathbb{T}$, allocate the item to a bidder who has the highest value at t and who has not been assigned an item yet (breaking ties deterministically). Every allocated bidder pays its critical value, which is collected upon its reported departure.*

Claim 1. *The allocation rule of Mechanism 2 satisfies (value, time, identifier)-monotonicity.*

Proof. Let us describe the allocation rule f_i^t for bidder i at period t when θ_{-i} is fixed:

$$f_i^t(\theta_i, \theta_{-i}) = \begin{cases} 1 & \text{if } a_i \le t \le d_i \text{ and unallocated in } t' < t \\ & \quad \text{and } r_i \ge r_l \ \forall l (\in \mathbb{N} \setminus \{i\}) \text{ s.t.} \\ & \quad\ a_l \le t \le d_l \text{ and unallocated in } t' < t \\ 0 & \text{otherwise.} \end{cases}$$

We derive a contradiction by assuming that, when there exist at least one winner j' in ϕ_i for some $\phi_i, \theta_{-\phi_i}, \theta_{\phi_i}$, there also exists type θ_i that satisfies Eq. (3).

Choose winner $j' \in \phi_i$ whose arrival period $a_{j'}$ is the earliest among the identifiers in ϕ_i. Let $t_{j'}$ denote the period in which j' wins. At period $t_{j'}$, θ_i is present, since $a_i \le a_{j'} \le d_{j'} \le d_i$ holds from Eq. (3). The bid $r_{j'}$ is the highest one at $t_{j'}$ in the presence of ϕ_i. Consider the absence of ϕ_i. Since ϕ_i is replaced with $\mathbf{0}$, the highest bid changes from $r_{j'}$ to r_i at $t_{j'}$. $r_i \ge r_l$ for all $l \in \mathbb{N} \setminus \phi_i$ such that bidder l is present at $t_{j'}$ ($a_l \le t_{j'} \le d_l$) and is unallocated in $t' < t_{j'}$. Thus, bidder i with type θ_i is chosen as a winner at period $t_{j'}$ (or before $t_{j'}$) if θ_{ϕ_i} is replaced with $(\theta_i, \mathbf{0}, \ldots, \mathbf{0})$. Accordingly, this contradicts the assumption. □

We can easily verify that the payment rule of Mechanism 2 is defined appropriately such that the mechanism is false-name-proof. We omit the proof due to space limitation.

4. NON-TRIVIAL FALSE-NAME-PROOF MECHANISMS

In this section, we present two non-trivial false-name-proof online mechanisms for k identical items. Before introducing them, we recall the intuitive reason why Mechanism 1 is not false-name-proof. The number of bidders, probably including false identifiers, determines when Mechanism 1 transits to the accepting phase. More precisely, a bidder can manipulate the transition period a^*, since it is set as the arrival period of the $\lfloor n/e \rfloor$-th bidder. To avoid such a manipulation, we must determine the transition period independently from the the number of bidders. The basic idea of our mechanisms is that they transit in a predefined constant period τ. The following is our first mechanism.

MECHANISM 3. *Let k be the number of items for sale and τ be a predefined period s.t. $0 \le \tau \le T$.*

1. *(learning phase): At period τ, sort the bidding values observed so far in descending order and denote them as $r_{(1)}, r_{(2)}, \ldots, r_{(k)}, r_{(k+1)}, \ldots$. If there exist only $k'(< k+1)$ bids, we assume $r_{(k'+1)}, \ldots$ are 0.*
2. *(transition): If any bidder who bids $r_{(1)}, \ldots, r_{(k)}$ is still present at period τ, then sell to that bidder at price $r_{(k+1)}$.*
3. *(accepting phase): As long as there exists a remaining item, sell to the next bidder whose bid is at least $r_{(k)}$ at price $r_{(k)}$.*

When $k = 1$, Mechanism 3 can be considered an application of the optimal stopping rule for a class of secretary problems, where the number of candidates n is unknown [3].

EXAMPLE 2. *Now let us describe the behavior of Mechanism 3 based on Table 1, assuming $\tau = 3$ and $k = 1$. First, consider the case where bidder 1 reports truthfully. From the definition, Mechanism 3 does not allocate the item to any bidder until period 3. At period 3, it allocates the item to bidder 1 at price 0. Next, consider another case where she uses two identifiers, $1'$ and $1''$, and reports $(1, 3, 6)$ and $(2, 2, \epsilon)$, respectively. Again, the mechanism does not allocate the item to any bidder until period 3. At period 3, it allocates the item to identifier $1'$ at price ϵ. Clearly that bidder 1 cannot increase her utility even if she uses false identifiers in Mechanism 3. Furthermore, losing bidders 2-6 cannot be winners even if they use false identifiers, since we assume the no-early arrival, no-late departure property.*

Although Mechanism 3 is false-name-proof, it requires a predefined transition period. In general, it is difficult to determine an appropriate transition period with respect to efficiency and revenue. However, we show the competitive ratio below in this section, assuming the mechanism knows the distribution of bidder arrival times.

On the other hand, one might think that the bidders who depart before the transition period do not have an incentive to join the auction, since they know that they have no chance to win. One possible remedy is to keep the information about the transition period τ private by not announcing it beforehand. Another remedy is to use a random timing device to determine the transition period. It can ring with small enough probability in each period before the default transition period τ and must ring in τ at the latest.

Utilizing our characterization, we show the next theorem.

THEOREM 2. *Mechanism 3 is false-name-proof.*

PROOF. We first prove that the allocation rule of this mechanism satisfies (value, time, identifier)-monotonicity and then show that the payment rule is defined by critical values. When θ_{-i} is fixed, the allocation rule f_i for bidder i can be described as follows. We denote the k-th highest value observed until τ except i's bid as $r^{-i}_{(k)}$.

$$f_i(\theta_i, \theta_{-i}) = \begin{cases} 1 & \text{if either (i) } a_i \le \tau \wedge d_i \ge \tau \wedge r_i \ge r^{-i}_{(k)}, \\ & \quad \text{or (ii) } a_i > \tau \wedge r_i \ge r^{-i}_{(k)} \wedge |W| < k, \\ & \quad \text{where } W = \{w \mid w \ne i \wedge a_w \le a_i \\ & \qquad \wedge d_w \ge \tau \wedge r_w \ge r^{-i}_{(k)}\} \\ 0 & \text{otherwise.} \end{cases}$$

We are going to derive a contradiction by assuming that the allocation rule does not satisfy (value, time, identifier)-monotonicity. More specifically, we assume that when at least one winner l exists in ϕ_i, for some $\phi_i, \theta_{-\phi_i}, \theta_{\phi_i}$, there exists type $\theta_i = (a_i, d_i, r_i)$ such that

$$\big(\forall j \in \phi_i, a_i \le a_j \le d_j \le d_i\big) \wedge r_i \ge \textstyle\sum_{j' \in \phi_i : j' \text{ wins}} r_{j'}$$
$$\text{and } f_i((\theta_i, \mathbf{0}, \ldots, \mathbf{0}), \theta_{-\phi_i}) = 0.$$

Choose j' as the winner in ϕ_i and its arrival period $a_{j'}$ is earliest. Note that j' is a false identifier owned by i.

First, consider the case where $a_i \le \tau$. Since j' is a winner, regardless whether $a_{j'}$ is before or after τ, $d_{j'} \ge \tau$ and $r_{j'} \ge r^{-j'}_{(k)}$. Also, $r^{-i}_{(k)} \le r^{-j'}_{(k)}$. This is because $r^{-i}_{(k)}$ is the k-th highest valuation observed until τ except the bid of i, and $r^{-j'}_{(k)}$ is the k-th highest valuation observed until τ, including ϕ_i except j'. >From the assumption, $a_i \le a_{j'} \le \tau$, $d_i \ge d_{j'} \ge \tau$, and $r_i \ge r_{j'}$. Thus, we obtain $r_i \ge r^{-j'}_{(k)} \ge r^{-i}_{(k)}$, and condition (i) of the allocation rule holds. This contradicts the assumption that $f_i((\theta_i, \mathbf{0}, \ldots, \mathbf{0}), \theta_{-\phi_i}) = 0$.

Next, consider the case where $a_i > \tau$. For all $j \in \phi_i$, $a_i \le a_j$ holds; no bidder in ϕ_i arrives before τ. Thus, $r^{-i}_{(k)} = r^{-j'}_{(k)}$ holds. Since j' is the winner, $r_{j'} \ge r^{-j'}_{(k)}$ also holds. From the assumption, $r_i \ge r_{j'}$ holds. Thus, we obtain $r_i \ge r^{-i}_{(k)}$. Also, since j' is the winner in ϕ_i and its arrival period is the earliest. Thus, for $W_{j'} = \{w \mid w \in \mathbb{N} \setminus \{j'\}$ and $a_w \le a_{j'}$ and $d_w \ge \tau$ and $r_w \ge r^{-j'}_{(k)}\}$ and $W_i = \{w \mid w \in \mathbb{N} \setminus \phi_i$ and $a_w \le a_i$ and $d_w \ge \tau$ and $r_w \ge r^{-i}_{(k)}\}$, since $a_i \le a_{j'}$, $W_i \subseteq W_{j'}$ holds. Thus, $|W_i| \le |W_{j'}|$ holds. Since j' is a winner, $|W_{j'}| < k$ holds. Thus, $|W_i| < k$ holds. Therefore, condition (ii) of the allocation rule holds, but this contradicts the assumption.

Critical value cv of bidder i is defined as follows:

$$\mathrm{cv}(a_i, d_i, \theta_{-i}) = \begin{cases} r^{-i}_{(k)} & \text{if either (i) } a_i \le \tau \wedge d_i \ge \tau, \\ & \quad \text{or (ii) } a_i > \tau \wedge |W| < k, \\ & \quad \text{where } W = \{w \mid w \ne i \\ & \qquad \wedge a_w \le a_i \wedge d_w \ge \tau \\ & \qquad \wedge r_w \ge r^{-i}_{(k)}\} \\ \infty & \text{otherwise.} \end{cases}$$

Also, the appropriate payment rule p is derived as follows:

$$p_i(\theta_i, \theta_{-i}) = \begin{cases} \mathrm{cv}(a_i, d_i, \theta_{-i}) & \text{if } f_i(\theta_i, \theta_{-i}) = 1 \\ 0 & \text{otherwise.} \end{cases}$$

This payment rule is identical to Mechanism 1. □

Competitive analysis for online mechanisms requires to assume an adversarial model as well as the optimal stopping theory. A representative model is the random-ordering model used in [9], which requires a mechanism to observe the exact number of bidders beforehand. Therefore, we cannot apply the model to our situation where a mechanism can not certainly observe the number of real bidders. Thus, we introduce another adversarial model from [3]. Unlike the random-ordering model, the model requires mechanisms to observe only the distribution of arrival times of bidders. It is quite natural that a mechanism has knowledge about the distribution of arrival times in such real-world economic environments as Internet auctions. For example, an auctioneer can usually obtain trends about the density of bids, e.g., the number of bids on weekends exceeds those in the daytime on weekdays. Focusing on the model where only the distribution of bidder arrival times is known, we apply Bruss's adversarial model to our competitive analysis. Notice that Hajiaghayi et al. [10] deal with a situation where the number of bidders is unknown to a mechanism. Instead, although they assume that the distribution of valuations of bidders is known, we will investigate this model in future work.

In our model, the auction is performed within finite continuous interval $[0,T]$ and all bidders are *impatient*; $\forall i \in N$, $a_i = d_i$. This continuous model makes the analysis much simpler; in a discrete time interval model, there might be no transition period τ s.t. $G(\tau) = \frac{1}{2}$ in Theorem 3. We assume that for n bidders, an adversary specifies its valuations. We also restrict our attention to cases where all valuations are unique to ignore ties. In addition, we let a mechanism know a distribution function G, from which bidder arrival times are drawn i.i.d. However, the mechanism has neither information about the number of bidders n nor their valuations.

For $k = 1$, our model becomes almost identical to that of the secretary problem discussed in [3]. Thus, we can easily see that for any distribution G, Mechanism 3 is e-competitive for efficiency by defining $\tau = G^{-1}(e^{-1})$. Although we strongly believe this stopping rule is optimal for efficiency, we cannot directly use the result in [3], since the mechanism can observe richer information, i.e., the bids of bidders, than for the secretary problem. However, as discussed in [9], it is very unlikely that a mechanism can capitalize on this numerical information, since we are making absolutely no assumptions about the distribution of bids.

We now show a more general result for arbitrary k. The next theorem shows that the competitive ratio of Mechanism 3 for efficiency is independent of the number of items k, if the transition period τ satisfies $F(\tau) = \frac{1}{2}$.

THEOREM 3. *In our model, Mechanism 3 with constant stopping time $\tau_{1/2}$ such that $G(\tau_{1/2}) = \frac{1}{2}$ is 4-competitive for efficiency as $n \to \infty$ when k is sufficiently large and all bidders are impatient in finite continuous interval $[0,T]$.*

PROOF (SKETCH). In the worst case, each of the top k bidders has a high value (e.g., 1) and the others have a low value (e.g., 0). The probability that k-th highest bidder, who arrives before $\tau_{1/2}$, is $k+s+1$-st highest overall, is given as $\binom{k+s}{k-1} \cdot (\frac{1}{2})^{k+s+1}$. Possible winners are bidders $1,\ldots,k+s$. A winner must arrive after $\tau_{1/2}$ and before k items are sold out. The actual value of efficiency (i.e., the expected number of winners $1,\ldots,k$) is given as $SS = \sum_{s=0}^{\infty} g(s) \cdot \min(s+1, k)$, where $g(s) = \binom{k+s}{k-1} \cdot (\frac{1}{2})^{k+s+1} \cdot \frac{k}{k+s}$. Clearly, this is smaller than $SS' = \sum_{s=0}^{\infty} g(s) \cdot k$, which equals $\frac{k^2}{k-1}(\frac{1}{2} - \frac{1}{2^k})$ by multinomial coefficient. Thus, for sufficiently larger k, $SS' > k/2$ holds. Furthermore, we can prove that $SS' \leq 2SS$ holds; SS' is an over-estimation of SS but SS' is at most twice as large as SS. More specifically, the amount of over-estimation, i.e., $SS' - SS$ is given as $\sum_{s=0}^{k-2} g(s) \cdot (k-s-1)$. We can show that this is smaller than SS, i.e., $SS' - SS \leq SS$ holds, since $g(s)$ is basically an increasing function of s (where s is smaller than $k-2$). Thus, $SS > k/4$ holds. Since the optimal social surplus is k, we obtain the competitive ratio of 4. □

In contrast to efficiency, the competitive ratio of Mechanism 3 for revenue is 0, which occurs in the same valuations above. To achieve better revenue, we introduce another mechanism.

MECHANISM 4. *Let k be the number of items for sale and $\tau_1,\ldots,\tau_k$ ($\tau_1 < \ldots < \tau_k$) be a sequence of predefined periods.*

1. *(learning phase): At period τ_m ($1 \leq m \leq k$), sort bidding values observed so far in descending order and denote them as $r^m_{(1)}, r^m_{(2)}, \ldots$. If there exist no bids, we assume $r^m_{(1)}, r^m_{(2)}, \ldots$ are 0.*
2. *(transition): If the bidder of $r^m_{(1)}$ is still present at period τ_m, then sell to him at price $r^m_{(2)}$.*
3. *(accepting phase): As long as the item remains and current time t satisfies $t < \tau_{m+1}$, sell to the next bidder whose bid is at least $r^m_{(1)}$ at price $r^m_{(1)}$.*

Intuitively, Mechanism 4 is false-name-proof, since the prices at transition periods $\tau_1,\ldots,\tau_k$ never decrease, and an unsold item will not be carried forward to the next period.

THEOREM 4. *In our model, Mechanism 4 with a sequence of stopping times $\tau_1,\ldots,\tau_k$ s.t., $G(\tau_m) = \frac{mT}{k+1}$ $\forall m \in \{1,\ldots,k\}$ is $\frac{k}{\log k}$-competitive for revenue as $n \to \infty$ when k is sufficiently large and all bidders are impatient in finite continuous interval $[0,T]$.*

PROOF. An adversary chooses a set of valuations so that all bidders $i \in \{1,\ldots,k\}$ have $1 - i \cdot \epsilon$ and all other $n-k$ bidders $i \in k+1,\ldots,n$ have $(n-i+1)\cdot\epsilon$ as the worst case.

The probability that a particular pair of bidders arrives within the same period is $\frac{1}{k}$. For sufficiently large k, the probability becomes small enough to be ignored. The probability that bidder 1 wins an item and pays a high value is given by the summation of the probabilities that bidder 1 arrives (a) after bidder 2, (b) before bidder 2 and after bidder 3, (c) before bidders 2 and 3 and after bidder 4, ..., i.e., $\frac{1}{2} + \frac{1}{3!} + \frac{2!}{4!} + \cdots + \frac{(k-2)!}{k!} = 1 - \frac{1}{k}$. In general, the probability that bidder i wins and pays a high value is $\frac{(i-1)!}{(i+1)!} + \cdots + \frac{(k-2)!}{k!} = \frac{1}{i} - \frac{1}{k}$. Thus, the expected revenue is calculated as $\sum_{i=1}^{k-1} \frac{1}{i} - \frac{k-1}{k}$. Since the first term $\sum_{i=1}^{k-1} \frac{1}{i}$ is a harmonic series, we have $\sum_{i=1}^{k-1} \frac{1}{i} - \frac{k-1}{k} \geq \log k - \frac{k-1}{k}$ and for large k, Mechanism 4 is $\frac{k}{\log k}$-competitive. □

Using a similar argument to the above proof, we can also show that Mechanism 4 is $\frac{k}{\log(k+1)}$-competitive for efficiency.

The competitive ratios shown in Theorems 4 and 5 are not tight since, we have not yet obtained theoretical lower bounds. However, even in one-shot mechanisms, there have been very few results on the competitive ratios of false-name-proof mechanisms, except for those by [11, 7]. Thus, we believe the results in this paper are an important first step to clarify the bounds in online false-name-proof mechanisms.

5. EXPERIMENTAL ANALYSIS

In addition to the worst-case analysis in Section 4, we experimentally evaluated Mechanism 3 when $k = 1$. We set discrete time periods $\{1,\ldots,20\}$ ($T = 20$), varying the number of bidders from 10 to 100 by 10. Each bidder's type $\theta_i = (a_i, d_i, r_i)$ is generated as follows. The valuation r_i is drawn from a uniform distribution over $[0, \bar{r}]$. The arrival time a_i is drawn from a uniform distribution over $[0,T]$, and the departure time d_i is drawn from a uniform distribution over $[a_i, T]$. Notice that, although we run our simulation with a variety of values $\bar{r}$, the performance does not depend on $\bar{r}$. Thus, we show the results in the case of $\bar{r} = 100$. We set the stopping strategy τ of Mechanism 3 to $\lfloor T/e \rfloor = 7$.

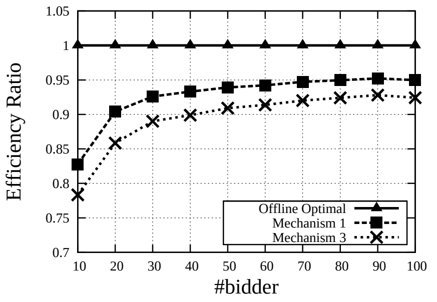


Figure 2: Efficiency ratios in average case with respect to Offline Optimal Mechanism

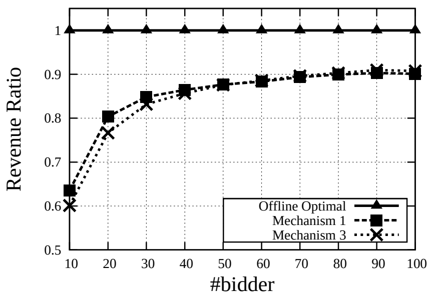


Figure 3: Revenue ratios in average case with respect to Offline Optimal Mechanism

From Theorem 3, this is the stopping strategy to achieve a competitive ratio of e for efficiency if the bidders are impatient. We then averaged the ratios of efficiency and revenue, generating 10000 instances for each number of bidders.

Figures 2 and 3 illustrate the average ratios of efficiency and revenue, respectively, achieved by Mechanism 1 and Mechanism 3, varying the number of bidders. Note that the result of Mechanism 1 is provided to show an ideal ratio, where the mechanism can set the optimal learning period by knowing the number of bidders n beforehand, and bidders do not use false-name bids. In Fig. 2, we can see that in terms of efficiency, Mechanism 3 achieves 93% of the offline optimal mechanism as the number of bidders grows and it is slightly outperformed by Mechanism 1. Furthermore, in terms of revenue, Fig. 3 shows that Mechanism 3 performs almost equivalently to Mechanism 1.

6. CONCLUSIONS AND FUTURE WORKS

In this paper, we characterized false-name-proof online mechanisms and proposed two non-trivial ones for k identical items. When $k = 1$, Mechanism 3 corresponds to the optimal stopping rule of a class of secretary problems [3], where the number of candidates n is unknown to the employer who only knows the distribution of the candidate arrival times. We further revealed that Mechanism 3 is 4-competitive for efficiency, which is independent on the number of items k. Also, Mechanism 4 is $\frac{k}{\log k}$-competitive for revenue.

One open problem is obtaining a lower bound of the competitive ratio of false-name-proof online mechanisms for efficiency and revenue. For efficiency, we strongly believe that the lower bound is e when $k = 1$, although it remains unproved. We would like to relax several assumptions we introduced for competitive analysis, e.g., impatient bidders. Furthermore, we would like to extend our results beyond single-valued domains (e.g., dynamic multi-unit auctions [5]). Considering the case that a bidder can only use a limited number of fake identifiers might also be interesting. This restriction would weaken bidders in the market, and help us design false-name-proof mechanisms.

7. ACKNOWLEDGMENTS

This paper is partially supported by KAKENHI (20240015 and 23500184) and Grant-in-Aid for JSPS research fellows.

8. REFERENCES

[1] Y. Bachrach and E. Elkind. Divide and conquer: false-name manipulations in weighted voting games. In *AAMAS*, 2008.

[2] S. Bikhchandani, S. Chatterji, R. Lavi, A. Mu'alem, N. Nisan, and A. Sen. Weak monotonicity characterizes deterministic dominant-strategy implementation. *Econometrica*, 74(4):1109–1132, 2006.

[3] F. Bruss. A unified approach to a class of best choice problems with an unknown number of options. *The Annals of Probability*, 12(3):882–889, 1984.

[4] V. Conitzer and M. Yokoo. Using mechanism design to prevent false-name manipulations. *AI Magazine*, 31(4):65–78, 2010.

[5] E. H. Gerding, V. Robu, S. Stein, D. C. Parkes, A. Rogers, and N. R. Jennings. Online mechanism design for electric vehicle charging. In *AAMAS*, 2011.

[6] A. V. Goldberg, J. D. Hartline, A. R. Karlin, M. Saks, and A. Wright. Competitive auctions. *Games and Economic Behavior*, 55(2):242 – 269, 2006.

[7] M. Guo and V. Conitzer. False-name-proofness with bid withdrawal. In *AAMAS*, 2010.

[8] M. T. Hajiaghayi, R. D. Kleinberg, M. Mahdian, and D. C. Parkes. Online auctions with re-usable goods. In *EC*, 2005.

[9] M. T. Hajiaghayi, R. D. Kleinberg, and D. C. Parkes. Adaptive limited-supply online auctions. In *EC*, 2004.

[10] M. T. Hajiaghayi, R. D. Kleinberg, and T. Sandholm. Automated online mechanism design and prophet inequalities. In *AAAI*, 2007.

[11] A. Iwasaki, V. Conitzer, Y. Omori, Y. Sakurai, T. Todo, M. Guo, and M. Yokoo. Worst-case efficiency ratio in false-name-proof combinatorial auction mechanisms. In *AAMAS*, 2010.

[12] R. Lavi and N. Nisan. Competitive analysis of incentive compatible on-line auctions. In *EC*, 2000.

[13] R. B. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981.

[14] D. C. Parkes. Online mechanisms. In N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*, chapter 16, 2007.

[15] T. Todo, A. Iwasaki, M. Yokoo, and Y. Sakurai. Characterizing false-name-proof allocation rules in combinatorial auctions. In *AAMAS*, 2009.

[16] M. Yokoo, Y. Sakurai, and S. Matsubara. The effect of false-name bids in combinatorial auctions: New fraud in internet auctions. *Games and Economic Behavior*, 46(1):174–188, 2004.

Session 1E
Game Theory I

Existence of Stability in Hedonic Coalition Formation Games

Haris Aziz
Department of Informatics
Technische Universität München
85748 Garching bei München, Germany
aziz@in.tum.de

Florian Brandl
Department of Mathematics
Technische Universität München
85748 Garching bei München, Germany
f.brandl@mytum.de

ABSTRACT

In this paper, we examine *hedonic coalition formation games* in which each player's preferences over partitions of players depend only on the members of his coalition. We present three main results in which restrictions on the preferences of the players guarantee the existence of stable partitions for various notions of stability. The preference restrictions pertain to *top responsiveness* and *bottom responsiveness* which model optimistic and pessimistic behavior of players respectively. The existence results apply to natural subclasses of *additively separable hedonic games* and *hedonic games with $\mathcal{B}$-preferences.* It is also shown that our existence results cannot be strengthened to the case of stronger known stability concepts.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; J.4 [**Computer Applications**]: Social and Behavioral Sciences - Economics

General Terms

Economics, Theory and Algorithms

Keywords

Game theory (cooperative and non-cooperative), teamwork, coalition formation, and coordination

1. INTRODUCTION

In many models of multiagent interaction such as roommate matching and exchange of discrete goods, deviations from one outcome to another can cycle and it may well be possible that no stable outcome is guaranteed. This leads to one of the most fundamental questions in game theory: *what are the necessary and sufficient conditions for the existence of stable outcomes?* This question has been examined extensively by researchers working in market design, multiagent systems, and operations research. We address this question in the context of coalition formation games in which outcomes are partitions of the players. We focus on *hedonic coalition formation games* in which each player's preferences over partitions depend only on the members of his coalition. Hedonic games are a rich and versatile class of coalition formation games which also encapsulate various stable matching scenarios [see e.g., 5, 6, 7, 14].

In game theory and multiagent systems, understanding the conditions under which systems and social outcomes are guaranteed to be in equilibrium is a fundamental research problem. In this paper, we advance the state of the art on existence results for hedonic games. We strengthen the recently introduced stability concept *strong Nash stability* [15] to *strict strong Nash stability* and show that *top responsiveness* and *mutuality*—conditions different from ones in [15]—are sufficient for the existence of a strictly strong Nash stable partition in any hedonic game. The result applies to natural subclasses of *additively separable hedonic games* [6]. It is also shown that top responsiveness and mutuality together do not guarantee the non-emptiness of the set of *perfect* partitions—a natural concept stronger than strict strong Nash stability.

We then consider a recently introduced property of hedonic games called *bottom refuseness* [17] which we will refer to as *bottom responsiveness.* A new stability notion called *strong individual stability* is formulated which is stronger than both core stability and individual stability. It is shown that bottom responsiveness guarantees the existence of a strong individually stable partition. Also, the combination of *strong bottom responsiveness* and mutuality guarantees the existence of a strong Nash stable partition. Our results concerning bottom responsive games cannot be strengthened to any stronger known stability concept. They also apply to *'aversion to enemies' games* introduced in [11].

Outline.

In Section 2, we present the backdrop of our results and discuss related work. We then introduce hedonic games and the stability concepts considered for these games in Section 3. The relationships between the stability concepts are expounded and clarified in Section 4. We then proceed to Sections 5 and 6 in which the main results are presented. Section 5 concerns hedonic games satisfying top responsiveness whereas in Section 6, existence results concerning bottom responsive games are presented. In Section 7, well-studied subclasses of hedonic games such as additively separable hedonic games and hedonic games with $\mathcal{B}$-preferences are considered and it is shown how existence results apply to these games. Finally, we conclude the discussion in Section 8.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

2. RELATED WORK

Identifying sufficient and necessary conditions for the existence of stability in coalition formation has been active area of research. Perhaps the most celebrated result in this field is the existence of a (core) stable matching for the stable marriage problem via the Gale-Shapley algorithm [13]. Later, Banerjee et al. [5] proved that if a hedonic game satisfies a condition called *weak top coalition property*, then the core is non-empty. Banerjee et al. [5] also showed that for various restrictions over preferences, stability is still not guaranteed.

In another important paper, Bogomolnaia and Jackson [6] formalized Nash stability and individual stability in the context of hedonic games and presented a number of sufficient conditions for the existence of various stability concepts. For instance, they showed that *symmetric additively separable preferences* guarantee the existence of a Nash stable partition. A hedonic game is additively separable if each player has a cardinal value for every other player and the player's utility in a partition is the sum of his values for the players in his coalition. The strict core and core is also non-empty for *'appreciation of friends'* and *'aversion to enemies'* games respectively—two simple classes of additively separable games [11].

Alcalde and Revilla [1] proposed a natural preference restriction called top responsiveness which is based on the idea that players value other players on how they could complement them in research teams. They showed that there exists an algorithm called the *Top Covering Algorithm* which finds a core stable partition for top responsive hedonic games. The Top Covering Algorithm can be seen as a generalization of *Gale's Top Trading Cycle algorithm* [16]. Dimitrov and Sung [9, 10] simplified the Top Covering Algorithm and proved that top responsiveness implies non-emptiness of the strict core and if mutuality is additionally satisfied, then a Nash stable partition exists.

In a follow-up paper, Suzuki and Sung [17] introduced bottom refuseness in an analogous way to top responsiveness. They showed that for hedonic games satisfying bottom refuseness, the *Bottom Avoiding Algorithm* returns a core stable partition. Suzuki and Sung [17] noted that 'appreciation of friends' and 'aversion to enemies' games satisfy top responsiveness and bottom responsiveness respectively, thereby explaining the results in [11].

Very recently, Karakaya [15] proposed a new stability concept called *strong Nash stability* which is stronger than Nash stability and core stability combined. He showed that strong-Nash is non-empty if the *weak top choice property* (stronger than the weak top coalition property) is satisfied or if preferences are '*descending separable*'. We will prove three different results in which natural restrictions on the player preferences guarantee the existence of stable partitions where stability is strong Nash stability or its generalization or variant.

3. HEDONIC GAMES & STABILITY CONCEPTS

In this section, we review the terminology, notation, and concepts related to hedonic games.

Hedonic games.

A *hedonic coalition formation game* is a pair $(N, \succsim)$ where N is a set of players and $\succsim$ is a *preference profile* which specifies for each player $i \in N$ the preference relation $\succsim_i$, a reflexive, complete and transitive binary relation on set $\mathcal{N}_i = \{S \subseteq N : i \in S\}$. $S \succ_i T$ denotes that i strictly prefers S over T and $S \sim_i T$ that i is indifferent between coalitions S and T. A *partition* π is a partition of players N into disjoint coalitions. By $\pi(i)$, we denote the coalition in π which includes player i.

Stability Concepts.

We present the various stability concepts for hedonic games. Nash stability, strict core stability, Pareto optimality, core stability, and individual rationality are classic stability concepts. Individual stability was formulated in [6]. Strong Nash stability was introduced by Karakaya [15] and perfect partitions were considered in [2]. In this paper, we also introduce strict strong Nash stability and strong individual stability which imply strong Nash stability and core stability respectively.

- A partition π is *individually rational (IR)* if no player has an incentive to become alone, i.e., for all $i \in N$, $\pi(i) \succsim_i \{i\}$.
- A partition is *perfect* if each player is in one of his most preferred coalition [2].
- A partition is *Nash stable (NS)* if no player can benefit by moving from his coalition to another (possibly empty) coalition T.
- A partition is *individually stable (IS)* if no player can benefit by moving from his coalition to another existing (possibly empty) coalition T while not making the members of T worse off.
- A coalition $S \subseteq N$ *blocks* a partition π, if each player $i \in S$ strictly prefers S to his current coalition $\pi(i)$ in the partition π. A partition which admits no blocking coalition is said to be in the *core (C)*.
- A coalition $S \subseteq N$ *weakly blocks* a partition π, if each player $i \in S$ weakly prefers S to $\pi(i)$ and there exists at least one player $j \in S$ who strictly prefers S to his current coalition $\pi(j)$. A partition which admits no weakly blocking coalition is in the *strict core (SC)*.
- A partition π is *Pareto optimal (PO)* if there is no partition π' with $\pi'(j) \succsim_j \pi(j)$ for all players j and $\pi'(i) \succ_i \pi(i)$ for at least one player i.
- For partition π, $\pi' \neq \pi$ is called *reachable* from π by movements of players $H \subseteq N$, denoted by $\pi \xrightarrow{H} \pi'$, if $\forall i, j \in N \setminus H, i \neq j : \pi(i) = \pi(j) \Leftrightarrow \pi'(i) = \pi'(j)$.

  A subset of players $H \subseteq N, H \neq \emptyset$ *strong Nash blocks* π if a partition $\pi' \neq \pi$ exists with $\pi \xrightarrow{H} \pi'$ and $\forall i \in H : \pi'(i) \succ_i \pi(i)$.

  If a partition π is not strong Nash blocked by any set $H \subseteq N$, π is called *strong Nash stable (SNS)* [15].
- A subset of players $H \subseteq N, H \neq \emptyset$ *weakly Nash blocks* π if a partition $\pi' \neq \pi$ exists with $\pi \xrightarrow{H} \pi'$, $\forall i \in H : \pi'(i) \succsim_i \pi(i)$ and $\exists i \in H : \pi'(i) \succ_i \pi(i)$.

  A partition which admits no weakly Nash blocking coalition is said to satisfy *strict strong Nash stability (SSNS)*.

- A non-empty set of players $H \subseteq N$ is *strongly individually blocking* a partition π, if a partition π' exists such that:

  1. $\pi \xrightarrow{H} \pi'$ (as for SNS),
  2. $\forall i \in H : \pi'(i) \succ_i \pi(i)$, and
  3. $\forall j \in \pi'(i)$ for some $i \in H : \pi'(j) \succsim_j \pi(j)$.

  A partition for which no strongly individually blocking set exists is *strongly individually stable (SIS)*.[1]

$$\begin{tikzcd}
 & \text{Perfect} \arrow[d] & \\
 & \text{SSNS} \arrow[dl] \arrow[dr] & \\
\text{SNS} \arrow[d] \arrow[dr] & & \text{SC} \arrow[dl] \arrow[d] \\
\text{NS} \arrow[d] & \text{SIS} \arrow[dl] \arrow[dr] & \text{PO} \\
\text{IS} \arrow[dr] & & \text{C} \arrow[dl] \\
 & \text{IR} &
\end{tikzcd}$$

Figure 1: Inclusion relationships between stability concepts for hedonic games. For e.g, every NS partition is also IS. NS, SC, PO, C and IR are classic stability concepts. IS was formulated in [6]; SNS in [15]; and perfect partitions in [2]. We also introduce SSNS and SIS in this paper.

Depending on the context, we will utilize abbreviations like SIS, SNS, SSNS, IS etc. either for adjectives (for e.g. IS for individually stable) or for nouns (for e.g. IS for individual stability).

4. RELATIONS BETWEEN STABILITY CONCEPTS

In this section, we will explore and clarify the inclusion relationships between the stability concepts. The inclusion relationships between stability concepts are depicted in Figure 1.

Proposition 1. *Strict core stability implies strong individual stability which implies individual stability and also core stability.*

Proof. Strict core stability implies strong individual stability. Assume that a partition π is strict core stable but not strong individually stable. Then, there exists a coalition $S \subseteq N$ such that $S \notin \pi$ and each player in S is at least as happy as in π and one player in S is strictly happier than in π. But this means that π is not strict core stable.

Strong individual stability trivially implies individual stability.

Finally, we show that strong individual stability implies core stability. Assume that a partition π is strong individually stable but not core stable. Then there is a core deviating coalition S. But this would mean that each player $i \in S$ is strictly better off than in $\pi(i)$. But this means that π is not strong individually stable. This completes the proof. $\square$

Strong Nash stability as introduced by Karakaya [15] is quite a strong stability notion as seen by the following simple proposition.

Proposition 2. *Strong Nash stability implies Nash stability and also core stability.*

Furthermore, even if a partition is both strict core stable and Nash stable, it is not necessarily strong Nash stable.

Proof. The first statement follows from the definitions of the stability concepts and was already pointed out by Karakaya [15]. In fact, it can also easily be shown that SNS implies SIS. If a partition is SNS, then there is no strong Nash blocking set. This implies that there does not exist any strongly individually blocking set.

We now show that even if a partition is both strict core stable and Nash stable, it is not necessarily strong Nash stable. The following example shows a game, that admits a strict core and Nash stable partition but no strong Nash stable partition.

Let $(N, \succsim)$ be a game with $N = \{1, 2, 3, 4\}$ and the preference profile specified as follows:

$$\{1,2\} \succ_1 \{1,4\} \succ_1 \{1\} \succ_1 ...$$
$$\{2,3\} \succ_2 \{1,2\} \succ_2 \{2\} \succ_2 ...$$
$$\{3,4\} \succ_3 \{2,3\} \succ_3 \{3\} \succ_3 ...$$
$$\{1,4\} \succ_4 \{3,4\} \succ_4 \{4\} \succ_4 ...$$

It is easy to check, that the partitions $\pi = \{\{1,2\},\{3,4\}\}$ and $\pi' = \{\{1,4\},\{2,3\}\}$ are both (even strictly) core stable and Nash stable. But neither of them is strong Nash stable since $\{2,4\}$ is blocking π and π' is blocked by $\{1,3\}$. Obviously any partition containing a coalition with 3 or more players is not even Nash stable, since each player prefers being alone to any coalition with more than 2 players. Also $\{\{1,3\},\{2,4\}\}$ is not even Nash stable, since it is not individually rational.

Nash and core stability prevent single players from moving to another (possibly empty) coalition or several players forming a new coalition respectively. In the given partitions π and π', it is possible for a pair of players to improve by switching coalitions and therefore prevent π and π' from being strong Nash stable. $\square$

In the next proposition, we show that although strong Nash stability is a strong stability concept, it implies neither strict core stability nor Pareto optimality.

Proposition 3. *Strict strong Nash stability implies strong Nash stability, strict core stability and Pareto optimality.*

On the other hand, strong Nash stability implies neither strict core stability nor Pareto optimality.

[1]SIS is a natural intermediate stability concept which is implied by strong Nash stability and strict core stability respectively and it also implies individual stability and core stability.

PROOF. Strict strong Nash stability trivially implies strong Nash stability. Strict strong Nash stability also implies strict core stability. If a partition is strong Nash stable, there exists no new coalition H, in which each player at least as happy and one player is strictly better off. Therefore, the partition is also strict core stable.

Now, we will show that strong Nash stability implies neither strict core stability nor Pareto optimality. Since, it is well-known that strict core stability implies Pareto optimality, it is sufficient to show that strong Nash stability does not imply Pareto optimality.

Strong Nash stability does not imply Pareto optimality. Consider the following four-player hedonic game:

$$\{1,2\} \sim_1 \{1,3\} \sim_1 \{1,4\} \succ_1 \cdots$$
$$\{1,2\} \sim_2 \{2,3\} \sim_2 \{2,4\} \succ_2 \cdots$$
$$\{2,3\} \sim_3 \{3,4\} \succ_3 \cdots$$
$$\{1,4\} \sim_4 \{2,4\} \succ_4 \{3,4\} \succ_4 \cdots$$

Then, the partition $\{\{1,2\},\{3,4\}\}$ is strong Nash stable. However it is Pareto dominated by $\{\{2,3\},\{1,4\}\}$. $\square$

In the next sections, we will present the central results of the paper.

5. TOP RESPONSIVENESS

Top responsiveness [1, 9, 10] and bottom responsiveness [17] are natural restrictions that are imposed on the individual preferences and not on the whole preference profile. The idea is that a player's preference for a coalition depends on the best and worst subcoalitions respectively. In this section, we present a result that a partition fulfilling SSNS exists for hedonic games satisfying top responsiveness and an additional property called mutuality (with respect to top responsiveness).

Top responsiveness.

Top responsiveness is based on *choice sets*—sets of players which each player wants to be with. Let $Ch(i,S)$—the *choice sets* of player i in coalition S—be defined as follows:

$$Ch(i,S) = \{S' \subseteq S : (i \in S') \wedge (S' \succsim_i S'' \; \forall S'' \subseteq S)\}.$$

A game satisfies *top responsiveness* if for each $i \in N$, the following three conditions hold:

1. for each $X \in \mathcal{N}_i$, $|Ch(i,X)| = 1$, (we denote by $ch(i,X)$ the unique maximal set of player i on X under $\succsim_i$),
2. for each pair $X, Y \in \mathcal{N}_i$, $X \succ_i Y$ if $ch(i,X) \succ_i ch(i,Y)$;
3. for each pair $X, Y \in \mathcal{N}_i$, $X \succ_i Y$ if $ch(i,X) = ch(i,Y)$ and $X \subset Y$.

A hedonic game satisfying top responsiveness additionally satisfies *mutuality* if

$$\forall i,j \in N, X \in \mathcal{N}_i \cap \mathcal{N}_j : i \in ch(j,X) \Leftrightarrow j \in ch(i,X).$$

We will also specify similar notion of mutuality with respect to hedonic games satisfying strong bottom responsiveness. When the context is clear, we will refer to the condition simply as mutuality.

EXAMPLE 1. *Let $(N,\succsim)$ be a game with $N = \{1,2,3\}$ and the preference profile specified as follows:*

$$\{1,2\} \succ_1 \{1,2,3\} \succ_1 \{1\} \succ_1 \{1,3\}$$
$$\{1,2,3\} \succ_2 \{1,2\} \sim_2 \{2,3\} \succ_2 \{2\}$$
$$\{2,3\} \succ_3 \{1,2,3\} \succ_3 \{3\} \succ_3 \{1,3\}$$

Then, $(N,\succsim)$ satisfies top responsiveness and mutuality.

We are now in a position to present our first result.

THEOREM 1. *Top responsiveness and mutuality together guarantee the existence of an SSNS partition.*

We prove Theorem 1 by showing that if a hedonic game satisfies top responsiveness and mutuality, then the Top Covering Algorithm of [1, 9, 10] returns an SSNS partition. Therefore, we identify conditions different than the ones identified by Karakaya [15] for which strong Nash stability is guaranteed. Since SSNS is stronger than SNS (Proposition 3) which in turn is stronger than even the combination of Nash stability and strict core stability (Proposition 2), Theorem 1 simultaneously strengthens the result in [9] and [10] in which it was shown that top responsiveness and mutuality together guarantee the existence of a Nash stable and strict core partition.

It can also be proved that Theorem 1 is optimal in the sense that it does not extend to perfect partitions. To be precise, we show that top responsiveness and mutuality together do not guarantee the existence of a perfect partition.

PROPOSITION 4. *Top responsiveness and mutuality together do not guarantee the existence of a perfect partition.*

PROOF. By counter example. In the game in Example 1, top responsiveness and mutuality are satisfied but no perfect partition exists. $\square$

Now that we have stated Theorem 1 and its complementing Proposition 4, we will present the proof of Theorem 1.

Proof of Theorem 1.

Firstly, we need additional definitions and a description of the Top Covering Algorithm. For each $X \subseteq N$, we denote by $\backsim_X$ the relation on $X \times X$ where $i \backsim_X j$ if and only if $j \in ch(i,X)$. In this case j is called a *neighbor* of i in X. Note that in the preference profiles satisfies top responsiveness mutuality, then $\backsim_X$ is a symmetric relation.
The connected component $CC(i,X)$ of i with respect to X is defined as follows:

$$CC(i,X) = \{k \in X : \exists j_1,\ldots,j_l \in X : i = j_1 \backsim_X \cdots \backsim_X j_l = k\}.$$

If $j \in CC(i,X)$, j is called *reachable* from i in X. Also note that $CC(j,X) \subseteq CC(i,X)$ if j is reachable from i and if mutuality is satisfied, then the following holds: $\forall X \subseteq N$, $i,j \in X : i \in CC(j,X) \Leftrightarrow j \in CC(i,X)$.

Now we are ready to present the simplified Top Covering Algorithm provided by Dimitrov and Sung [9, 10], adapted to the notation defined above. The algorithm is specified as Algorithm 1.

The following lemma will be used in the proof to Theorem 1.

Lemma 1. *Let $(N, \succsim)$ be a game satisfying top responsiveness and mutuality and π be the partition resulting by applying the simplified Top Covering Algorithm to it. Then*

$$\forall i \in N : ch(i, N) \subseteq \pi(i)$$

Proof. First we show by induction over the iterations of the algorithm that $ch(i, R^k) = ch(i, N)\ \forall i \in R^k,\ k = 1, 2, \ldots$. For $k = 1$, this is obviously true, because $R^1 = N$. Assume by induction, that $ch(i, R^k) = ch(i, N)\ \forall i \in R^k$. Let i' be the player selected in the k-th iteration of Step 3 and $j \in R^{k+1}$. Therefore $j \notin ch(i, R^k)\ \forall i \in CC(i', R^k)$. Because of mutuality $i \notin ch(j, R^k)\ \forall i \in CC(i', R^k)$. So $ch(j, R^k) \subseteq R^{k+1}$ and therefore $ch(j, R^{k+1}) = ch(j, R^k) = ch(j, N)$.

Now take an arbitrary player $i \in N$ and denote by k the iteration of the algorithm in which i was added to his coalition, i.e. $i \in S^k$. Let i' be the player selected in the k-th iteration of Step 3, so $i \in CC(i', R^k)$. Because of mutuality, $CC(i', R^k) = CC(i, R^k)$ and clearly $ch(i, R^k) \subseteq CC(i, R^k)$. From above, we know that $ch(i, N) = ch(i, R^k) \subseteq CC(i, R^k) = S^k = \pi(i)$. □

Algorithm 1 Top Covering Algorithm

```
Input: A hedonic game $(N, \succsim)$ satisfying top responsiveness.
1: $R^1 \leftarrow N$; $\pi \leftarrow \emptyset$.
2: for $k = 1$ to $|N|$ do
3:     Select $i \in R^k$ such that $|CC(i, R^k)| \leq |CC(j, R^k)|$ for
       each $j \in R^k$.
4:     $S^k \leftarrow CC(i, R^k)$; $\pi \leftarrow \pi \cup \{S^k\}$; and $R^{k+1} \leftarrow R^k \setminus S^k$
5:     if $R^{k+1} = \emptyset$ then
6:         return $\pi$
7:     end if
8: end for
9: return $\pi$
```

We note here that Lemma 1 may not hold, if mutuality is violated.

As shown by Dimitrov and Sung [9, 10] the resulting partition of the simplified Top Covering Algorithm is strict core stable as well as Nash stable if preferences as mutual. We are now ready to present the proof of Theorem 1.

Proof. Let π be the resulting partition and suppose it is not strictly strong Nash stable. Then a pair (H, π') exists where $H \subseteq N$ is the set of deviators and π' is the partition resulting after the deviation, i.e. $\pi \xrightarrow{H} \pi'$. Firstly, by Lemma 1, $ch(i, N) \subseteq CC(i, N)\ \forall i \in N$. Since H is a coalition blocking strict strong Nash stability, the following holds:

$$\forall i \in H : \pi'(i) \succsim_i \pi(i) \quad and$$
$$\exists j \in H : \pi'(j) \succ_j \pi(j).$$

Now consider the player j, who is better off in his new coalition $\pi'(j)$. Assume that $\pi(j) \cap \pi'(j) \subseteq H$, which means only deviators in $\pi(j) \cap \pi'(j)$. For $i \in \pi(j) \cap \pi'(j)$: $ch(i, \pi'(i)) \succsim_i ch(i, \pi(i))$, since $i \in H$ by assumption. We also know that $ch(i, \pi(i)) = ch(i, N)$ by Lemma 1. Therefore, for $i \in \pi(j) \cap \pi'(j)$: $ch(i, \pi'(i)) \succsim_i ch(i, \pi(i)) = ch(i, N)$. Because of uniqueness of choice sets in the definition of top responsiveness, $ch(i, \pi'(i)) = ch(i, N)$. So $ch(i, N) \subseteq \pi(i) \cap \pi'(i) = \pi(j) \cap \pi'(j)$.

$$\Longrightarrow \forall i \in \pi(j) \cap \pi'(j) : (\pi'(j) \cap \pi(j)) \succsim_i \pi'(j).$$

Due to assumption $\pi(j) \cap \pi'(j) \subseteq H$, the following holds:

$\forall i \in \pi(j) \cap \pi'(j) : (\pi(j) \cap \pi'(j)) \succsim_i \pi'(j) \succsim_i \pi(j) = \pi(i)$ & $(\pi(j) \cap \pi'(j)) \succsim_j \pi'(j) \succ_j \pi(j)$

So $\pi(j) \cap \pi'(j)$ would be a coalition blocking strict core stability, but Dimitrov and Sung [10] proved that π as produced by the simplified Top Covering Algorithm has to be strict core stable. Therefore $\pi'(j) \cap \pi(j) \nsubseteq H$ and there is at least one non-deviator in $\pi(j) \cap \pi'(j)$. Let us call this player i'.

Now take a look at the players in $\pi(j) \setminus \pi'(j)$. Note that this is not an empty set, because otherwise $\pi'(j) \supset \pi(j) \Longrightarrow \pi'(j) \precsim_j \pi(j)$. If one of them is not in H, then he was in the same coalition as i' in π, namely $\pi(j)$, and is now in a different, which is not consistent with $\pi \xrightarrow{H} \pi'$. So $(\pi(j) \setminus \pi'(j)) \subseteq H$. Because $\pi(j)$ is a connected component, at least one player k in $\pi(j) \setminus \pi'(j)$ has a friend l in $\pi'(j)$, meaning they are in each other's choice sets and as mentioned $l \in ch(k, N) \subseteq \pi(k)$. But now $l \notin \pi'(k)$ and therefore $\pi'(k) \prec_k \pi(k)$ which contradicts k being a deviator. □

6. BOTTOM RESPONSIVENESS

In this section, we present the central results concerning hedonic games which satisfy bottom responsiveness.

Bottom responsiveness.

Bottom responsiveness is a restriction on the preferences of each player in a hedonic game which models conservative or pessimistic agents. In contrast to top responsiveness, bottom responsiveness is based on *avoid sets*—sets of players which each player wants to avoid having in his coalition.

For any player $i \in N$ and $S \in \mathcal{N}_i$, $Av(i, S)$—the set of *avoid sets* of player i in coalition S—is defined as follows:

$$Av(i, S) = \{S' \subseteq S : (i \in S') \wedge (S' \precsim_i S''\ \forall S'' \subseteq S)\}.$$

A game satisfies *bottom responsiveness* if for each $i \in N$, the following conditions hold:

1. for each pair $X, Y \in \mathcal{N}_i$, $X \succ_i Y$ if $X' \succ_i Y'$ for each $X' \in Av(i, X)$ and each $Y' \in Av(i, Y)$; and
2. for each $i \in N$ and $X, Y \in \mathcal{N}_i$, $Av(i, X) \cap Av(i, Y) \neq \emptyset$ and $|X| \geq |Y|$ implies $X \succsim_i Y$.

A hedonic game $(N, \succsim)$ satisfies *strong bottom responsiveness* if it is bottom responsive and if for each $i \in N$ and $X \in \mathcal{N}_i$, $|Av(i, X)| = 1$. By $av(i, X)$, we denote the unique minimal set of player i on X under $\succsim_i$. The strong part of bottom responsiveness is analogous to Property 1 in the definition of top responsiveness. A hedonic game $(N, \succsim)$ satisfying strong bottom responsiveness additionally satisfies *mutuality* if for all $i, j \in N$, and X such that $i, j \in X$, $i \in av(j, X)$ if and only if $j \in av(i, X)$.

Example 2. *Let $(N, \succsim)$ be a game with $N = \{1, 2, 3\}$ and the preference profile specified as follows:*

$$\{1,3\} \succ_1 \{1\} \succ_1 \{1,2,3\} \succ_1 \{1,2\}$$
$$\{2,3\} \succ_2 \{2\} \succ_2 \{1,2,3\} \succ_2 \{1,2\}$$
$$\{1,2,3\} \succ_3 \{1,3\} \sim_3 \{2,3\} \succ_3 \{3\}$$

Then, $(N, \succsim)$ satisfies strong bottom responsiveness and also mutuality (with respect to strong bottom responsiveness).

For bottom responsive games, we prove that an SIS partition is guaranteed to exist even in the absence of mutuality.

THEOREM 2. *Bottom responsiveness guarantees the existence of an SIS partition.*

As a corollary, a core stable partition and an individually stable partition is guaranteed to exist. Previously, it was only known that the core is non-empty for bottom responsive games [17]. In contrast to the result by Suzuki and Sung [17], the proof of Theorem 2 does not require the Bottom Avoiding Algorithm. We associate with each IR partition a vector of coalition sizes in decreasing order. It is then shown via lexicographic comparisons between the corresponding vectors that arbitrary deviations between partitions are acyclic. With an additional natural constraint, even SNS is guaranteed (Theorem 3).

THEOREM 3. *Strong bottom responsiveness and mutuality together guarantee the existence of an SNS partition.*

We point out that Theorem 2 cannot be extended any further to take care of strict core stability and Theorem 3 cannot be extended to SSNS. The reason is that *symmetric 'aversion to enemies' games*—a subclass of strong bottom responsive games which satisfy mutuality—may not admit a strict core stable partition (Example 4, [11]).

Now that we have stated our results concerning hedonic games satisfying bottom responsiveness, we sketch the proofs.

Proof of Theorem 2.

For the use of further proofs, we introduce an ordering relation on the partitions. The definition will also apply to the proof of Theorem 3.

DEFINITION 1. *Let $N = \{1, ..., n\}$ be a set of players and π, π' two partitions of N, where $\pi = (S_1, ..., S_k)$ and $\pi' = (T_1, ..., T_l)$ with $|S_i| \geq |S_{i+1}|$ $\forall i \in \{1, ..., k-1\}$ and $|T_j| \geq |T_{j+1}|$ $\forall j \in \{1, ..., l-1\}$ respectively. We say, that*

$$\pi \dot{>} \pi' \Leftrightarrow \exists i \leq \min\{k, l\} : |S_i| > |T_i| \text{ and } |S_j| = |T_j| \ \forall j < i$$
$$\& \ \pi \doteq \pi' \Leftrightarrow k = l \text{ and } \forall i \leq k : |S_i| = |T_i|.$$

The relation $\dot{>}$ is complete, transitive and asymmetric, and places an ordering on the set of partitions. We now present the proof of Theorem 2 in which we utilize the relation $\dot{>}$.

PROOF. To simplify the presentation, we prove that every bottom responsive game admits an IS partition. The same argument can also be used to show that every bottom responsive game admits an SIS partition.

We show individual stability for each maximum element according to $\dot{>}$ of the set of individual rational coalitions. Consider the set $P = \{\pi' : \pi' \textit{ partitions } N \textit{ and } \forall S \in \pi', i \in S : \{i\} \in Av(i, S)\}$. Note that $P \neq \emptyset$, because the partition consisting of only singletons is in P and that P is a finite set because the number of partitions is finite. Denote by π a maximal element of P according to $\dot{>}$, i.e. $\pi \dot{\geq} \pi'$ $\forall \pi' \in P$. By definition, π is individually rational.

Now assume π is not individually stable. Then, there exists a player $i \in N$ and a coalition $S \in \pi \cup \{\emptyset\}$, such that $S \cup \{i\} \succ_i \pi(i)$ and $\forall j \in S : S \cup \{i\} \succsim_j S$. Now we show that the partition π resulting after the deviation of i is still individually rational and therefore an element of P. Clearly $S \neq \emptyset$ because of individual rationality of π. Furthermore $\{j\} \in Av(j, S \cup \{i\})$ $\forall j \in S$, because if not $S \cup \{i\} \prec_j S$ for some $j \in S$.
Consider a player $j \in \pi(i) \backslash \{i\}$. Due to individual rationality of π, $\{j\} \in Av(j, \pi(i))$, which implies $T \succsim_j \{j\}$ $\forall T \subseteq \pi(i)$ with $j \in T$. So $\pi(i) \setminus \{i\} \succsim_j \{j\}$. All other players $j \in N \setminus (\pi(i) \cup S)$ are not affected by the deviation of i because of the hedonic game setting. Therefore π' is individually rational and $\pi' \in P$.

The last step is to show $\pi' \dot{>} \pi$, which contradicts the maximality of π in P. Because player i improves by changing, $|S \cup \{i\}| > |\pi(i)|$ follows from condition 2) of bottom responsiveness . So $(S \cup \{i\}, \pi(i) \setminus \{i\}) \dot{>} (S, \pi(i))$ and all other coalitions are identical in π and π'. This contradicts $\pi \dot{\geq} \pi'$ and finishes the proof. $\square$

The proof also highlights a decentralized way to compute an IS or SIS partition. Start from the partition of singletons and enable arbitrary deviations. For each partition π_k, the new partition π_{k+1} is such that $\pi_{k+1} \dot{>} \pi_k$. Therefore, in a finite number of deviations, an IS or SIS partition is achieved.

Proof of Theorem 3.

We now present the proof of Theorem 3.

PROOF. We show strong Nash stability for each maximal element according to $\dot{\geq}$ of the set of individual rational coalitions (please see Definition 1). Consider the set $P = \{\pi' : \pi' \textit{ partitions } N \textit{ and } \forall S \in \pi', i \in S : \{i\} = av(i, S)\}$. Note that $P \neq \emptyset$, because the partition consisting of only singletons is in P and P is a finite set, because the number of partitions is finite. Denote by π a maximal element of P according to $\dot{\geq}$, i.e. $\pi \dot{\geq} \pi'$ $\forall \pi' \in P$.

Now assume π is not strong Nash stable. Then a set of players $H \subseteq N$ and a partition π' exist, such that

$$(1) \quad \pi \xrightarrow{H} \pi'$$
$$(2) \quad \forall i \in H : \pi'(i) \succ_i \pi(i).$$

We show that the partition π' resulting after the deviation is still individually rational and therefore an element of P. Clearly $av(i, \pi'(i)) = \{i\}$ $\forall i \in H$, because otherwise $\pi'(i) \succ_i \pi(i)$ would not hold. Now consider a player j such that $\pi'(j) \cap H \neq \emptyset$. $\forall i \in H \cap \pi'(j) : j \notin av(i, \pi'(j))$. Mutuality implies $i \notin av(j, \pi'(j))$ and therefore $av(j, \pi'(j)) = av(j, \pi(j)) = \{j\}$. All other players $j \in N$ are either not affected by any changes ($\pi(j) = \pi'(j)$) or they are left by some players in H ($\pi'(j) \subset \pi(j)$). In both cases $av(j, \pi'(j)) = av(j, \pi(j)) = \{j\}$, so π' is an element of P.

The last step is to show $\pi' \dot{>} \pi$, which contradicts the maximality of π in P. Because each player $i \in H$ improves, $|\pi'(i)| > |\pi(i)|$ $\forall i \in H$, which follows from condition (iii) of bottom responsiveness. Take the largest coalition $S \in \pi$ such that $S \cap H \neq \emptyset$. Obviously any coalition bigger than S in π at least does not get smaller after the deviation, because it contains no players from H. Then one of following two cases holds:

Case 1: at least one coalition $T \in \pi$ with $|T| > |S|$ gets joined by some player $i \in H$. But then $\pi' \dot{>} \pi$, since T increases in size and any larger coalition in π does not get smaller.

Case 2: If Case 1 does not hold, we know that no coalition in π larger than S is joined by a player in H and therefore stays the same. But one player $i \in S \cap H$ is part of a coalition $S' \in \pi'$ with $|S'| > |S|$. Since all coalitions in π, which are larger than S also exist in π', we can again conclude $\pi' \dot{>} \pi$.

In both cases $\pi' \dot{>} \pi$ which contradicts the maximality of π in P and finishes the proof. □

7. EXISTENCE OF STABILITY FOR SPECIFIC CLASSES OF GAMES

In this section, we highlight some natural subclasses of additively separable hedonic games [see e.g., 3, 6, 12, 14] and hedonic games with $\mathcal{B}$-preferences [see e.g., 8, 14] which guarantee top responsiveness or bottom responsiveness. Consequently, our existence results in Sections 5 and 6 and an existence result in the literature [10] applies to these settings.

Additively separable hedonic games.

Additively separable hedonic games are one of the most well-studied and natural class of hedonic games [see e.g., 3, 6, 12, 14]. In an *additively separable hedonic game (ASHG)* $(N, \succsim)$, each player $i \in N$ has value $v_i(j)$ for player j being in the same coalition as i and if i is in coalition $S \in \mathcal{N}_i$, then i gets utility $\sum_{j \in S \setminus \{i\}} v_i(j)$. For coalitions $S, T \in \mathcal{N}_i$, $S \succsim_i T$ if and only if $\sum_{j \in S \setminus \{i\}} v_i(j) \geq \sum_{j \in T \setminus \{i\}} v_i(j)$. Therefore an ASHG can be represented as (N, v). An ASHG is *symmetric* if $v_i(j) = v_j(i)$ for any two players $i, j \in N$ and is *strict* if $v_i(j) \neq 0$ for all $i, j \in N$.

We now formally introduce two classes of additively separable hedonic games which also satisfy top responsiveness and bottom responsiveness respectively. Both classes were introduced by Dimitrov et al. [11].

- An ASGH (N, v) is *appreciation of friends* if for all $i, j \in N$ such that $i \neq j$, the following holds: $v_i(j) \in \{-1, +n\}$.
- An ASGH (N, v) is *aversion to enemies* if for all $i, j \in N$ such that $i \neq j$, the following holds: $v_i(j) \in \{-n, +1\}$.

It is clear that 'appreciation of friends' and 'aversion to enemies' games are ASHGs with strict preferences. Suzuki and Sung [17] noted that 'appreciation of friends' and 'aversion to enemies' games satisfy top responsiveness and bottom responsiveness respectively. As a consequence, our main results apply to these games.

Corollary 1. *There exists an SSNS partition for each symmetric 'appreciation of friends' game.*

Proof. An 'appreciation of friends' game satisfies top responsiveness. Furthermore, if the game is (additively separable) symmetric, then it also satisfies mutuality with respect to top responsiveness. Then, as a result of Theorem 1, we get the corollary. □

Corollary 2. *There exists an SIS partition for each 'aversion to enemies' game.*

Proof. The statement follows from Theorem 2 and the fact that 'aversion to enemies' games satisfy bottom responsiveness. □

Corollary 3. *There exists an SNS partition for each symmetric 'aversion to enemies' game.*

Proof. It is already known that 'aversion to enemies' games satisfy bottom responsiveness. Since 'aversion to enemies' are additively separable hedonic games with strict preferences, they not only satisfy bottom responsiveness but also strong bottom responsiveness. If 'aversion to enemies' have symmetric preferences, then they not only satisfy strong bottom responsiveness but also (bottom responsive) mutuality. Therefore, we can apply Theorem 3 to derive the corollary. □

$\mathcal{B}$-hedonic games.

Finally, we show another important subclass of hedonic games called $\mathcal{B}$-hedonic games [8, 7] satisfies top responsiveness. In $\mathcal{B}$-hedonic games, players express preferences over players and these preferences over players are naturally extended to preferences over coalitions. We will assume that $\max_i(\emptyset) = \{i\}$. In *hedonic games with $\mathcal{B}$-preferences* (in short $\mathcal{B}$-hedonic games), for $S, T \in \mathcal{N}_i$, $S \succ_i T$ if and only if one of the following conditions hold:

1. for each $s \in \max_i(S \setminus \{i\})$ and $t \in \max_i(T \setminus \{i\})$, $s \succ_i t$, or
2. for each $s \in \max_i(S \setminus \{i\})$ and $t \in \max_i(T \setminus \{i\})$, $s \sim_i t$ and $|S| < |T|$.

A $\mathcal{B}$-hedonic has strict preferences for each $i \in N$ and $j, k \in N$, the following holds: $j \neq k \Rightarrow j \not\sim_i k$. Then, we have the following proposition.

Proposition 5. *$\mathcal{B}$-hedonic games with strict preferences satisfy top responsiveness.*

Proof. We show that $\mathcal{B}$-hedonic games with strict preferences satisfy all the three conditions of top responsiveness.

1. Firstly, for each $X \in \mathcal{N}_i$, $Ch(i, X) = \{\max_i X \cup \{i\}\}$ and thus $|Ch(i, X)| = 1$.
2. For a pair $X, Y \in \mathcal{N}_i$, assume that $ch(i, X) \succ_i ch(i, Y)$. This means that $\{\max_i(X)\} \cup \{i\} \succ_i \{\max_i Y\} \cup \{i\}$. Since the best player in X is more preferred by i than the best player in Y, then by the definition of $\mathcal{B}$-hedonic games, $X \succ_i Y$.
3. Finally, for each pair $X, Y \in \mathcal{N}_i$, assume that $ch(i, X) = ch(i, Y)$ and $X \subset Y$. Then, the player most preferred by i in X is the same as the player player most preferred by i in Y. Therefore, by the definition of $\mathcal{B}$-hedonic games, $X \succ_i Y$.

This completes the proof. □

Therefore, as a corollary we get the following statement which was proved by Cechlárová and Romero-Medina [8].

Corollary 4. *For each $\mathcal{B}$-hedonic game with strict preferences, a strict core stable partition is guaranteed to exist.*

PROOF. Dimitrov and Sung [10] showed that for hedonic games satisfying top responsiveness admit a strict core stable partition. Since $\mathcal{B}$-hedonic games satisfy top responsiveness, they admit a strict core stable partition. □

It will be interesting to see whether there are any natural restrictions on $\mathcal{B}$-hedonic games with strict preferences such that not only top responsiveness is satisfied but also (top responsive) mutuality is satisfied. In that case, we can apply Theorem 1 concerning SSNS to $\mathcal{B}$-hedonic games.

Our demonstrated connection between $\mathcal{B}$-hedonic games and top responsiveness goes deeper. The essential fact behind previous results concerning $\mathcal{B}$-hedonic games with strict preferences is that they satisfy top responsiveness. It turns out that the Top Covering Algorithm in [1] generalizes the B-STABLE algorithm in [8] and in fact Theorems 4.4 and 5.2 in [1] imply Theorem 1 and Theorem 2 in [8] respectively. This connection seems to have been unnoticed in the literature.

8. CONCLUSIONS

To conclude, we tried to paint a clearer picture of the landscape of stability concepts used in coalition formation games. The concepts ranged from standard ones such as the core to recently introduced concepts such as strong Nash stability. The core and strong Nash stability were generalized to strong individual stability and strict strong Nash stability respectively. The basic inclusion relationships between the stability concepts are depicted in Figure 1. Since hedonic games generalize various matching settings, the relations between the stability concepts also hold in matching settings such as two-sided matching, roommate matching etc.

We then examined restrictions on the preferences of agents which guarantee stable outcomes for the new stability concepts. Three main existence results (Theorems 1, 2 and 3) pertaining to top responsiveness and bottom responsiveness were presented. Our results strengthen or complement a number of results in the literature. We also showed that none of our existence results can be extended to a stronger known stability concept. It was seen that the theorems apply to some natural subclasses of hedonic games which have already been of interest among game-theorists. It will be interesting to find further applications of our existence results.

Identifying the impact of preference restrictions on stability also has algorithmic consequences. Recently, hedonic games have attracted research from an algorithmic and computational complexity point of view. There are various algorithmic questions such as checking the existence of and computing stable partitions for different representations of hedonic games (see e.g., [7, 14]). A general framework of preference restrictions and their impact on stability of partitions promises to be useful in devising generic algorithmic techniques to compute stable partitions. For example, we noted that the Top Covering Algorithm in [1] generalizes the B-STABLE algorithm in [8] by utilizing the insight that $\mathcal{B}$-hedonic games with strict preferences satisfy top responsiveness. We also mention the following interesting algorithmic questions. For hedonic games represented by individually rational lists of coalitions [4], what is the computational complexity of testing whether the game satisfies top reponsiveness or bottom responsiveness?

Our focus in the paper has been on sufficient conditions which guarantee the existence of stable outcomes. It will be interesting to see what additional conditions are required to ensure uniqueness of stable partitions for different notions of stability. Finally, characterizing the conditions for the existence of stability remains an open problem.

REFERENCES

[1] J. Alcalde and P. Revilla. Researching with whom? Stability and manipulation. *Journal of Mathematical Economics*, 40(8):869–887, 2004.

[2] H. Aziz, F. Brandt, and P. Harrenstein. Pareto optimality in coalition formation. In G. Persiano, editor, *Proceedings of the 4th International Symposium on Algorithmic Game Theory (SAGT)*, Lecture Notes in Computer Science (LNCS), pages 93–104. Springer-Verlag, 2011.

[3] H. Aziz, F. Brandt, and H. G. Seedig. Optimal partitions in additively separable hedonic games. In T. Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 43–48. AAAI Press, 2011.

[4] C. Ballester. NP-completeness in hedonic games. *Games and Economic Behavior*, 49(1):1–30, 2004.

[5] S. Banerjee, H. Konishi, and T. Sönmez. Core in a simple coalition formation game. *Social Choice and Welfare*, 18:135–153, 2001.

[6] A. Bogomolnaia and M. O. Jackson. The stability of hedonic coalition structures. *Games and Economic Behavior*, 38(2):201–230, 2002.

[7] K. Cechlárová. Stable partition problem. In *Encyclopedia of Algorithms*, pages 885–888. Springer, 2008.

[8] K. Cechlárová and A. Romero-Medina. Stability in coalition formation games. *International Journal of Game Theory*, 29:487–494, 2001.

[9] D. Dimitrov and S. C. Sung. Top responsiveness and Nash stability in coalition formation games. *Kybernetika*, 42(4):453–460, 2006.

[10] D. Dimitrov and S. C. Sung. On top responsiveness and strict core stability. *Journal of Mathematical Economics*, 43(2):130–134, 2007.

[11] D. Dimitrov, P. Borm, R. Hendrickx, and S. C. Sung. Simple priorities and core stability in hedonic games. *Social Choice and Welfare*, 26(2):421–433, 2006.

[12] M. Gairing and R. Savani. Computing stable outcomes in hedonic games with voting-based deviations. In *Proceedings of the 10th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 559–566, 2011.

[13] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[14] J. Hajduková. Coalition formation games: A survey. *International Game Theory Review*, 8(4):613–641, 2006.

[15] M. Karakaya. Hedonic coalition formation games: A new stability notion. *Mathematical Social Sciences*, 2011.

[16] L. S. Shapley and H. Scarf. On cores and indivisibility. *Journal of Mathematical Economics*, 1(1):23–37, 1974.

[17] K. Suzuki and S. C. Sung. Hedonic coalition formation in conservative societies. Technical Report 1700921, Social Science Research Network, 2010.

Stability Scores: Measuring Coalitional Stability

Michal Feldman
Harvard University and Hebrew University of Jerusalem[*]
mfeldman@seas.harvard.edu

Reshef Meir
Microsoft Research and Hebrew University of Jerusalem
reshef.meir@mail.huji.ac.il

Moshe Tennenholtz
Microsoft Research and Technion-Israel Institute of Technology
moshet@microsoft.com

ABSTRACT

We introduce a measure for the level of stability against coalitional deviations, called *stability scores*, which generalizes widely used notions of stability in non-cooperative games. We use the proposed measure to compare various Nash equilibria in congestion games, and to quantify the effect of game parameters on coalitional stability. For our main results, we apply stability scores to analyze and compare the Generalized Second Price (GSP) and Vickrey-Clarke-Groves (VCG) ad auctions. We show that while a central result of the ad auctions literature is that the GSP and VCG auctions implement the same outcome in one of the equilibria of GSP, the GSP outcome is far more stable. Finally, a modified version of VCG is introduced, which is group strategy-proof, and thereby achieves the highest possible stability score.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*;
J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*

General Terms

Algorithms, Theory, Economics

Keywords

Game theory, Auctions, Mechanism design, Solution concepts

1. INTRODUCTION

One of the most basic questions of game theory is: given a game in strategic form, what is its solution? By *solution* we typically mean a strategy profile that can be proposed to all agents, and no rational agent would want to deviate from it. Thus a solution should be *stable*. Many solution concepts for games have been studied; these studies differ by the level and interpretation of stability, as well as by the underlying assumptions that are required to achieve it. The best known solution concept for games is the Nash equilibrium (NE), a strategy profile from which no agent has an incentive to deviate *unilaterally*.

[*]At the time of research the author was affiliated with Microsoft Research in Herzlia, Israel.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

A basic problem with the NE solution concept is that a NE does not take into account joint deviations by coalitions of players. We usually assume that an individual will deviate from a profile if she has an available strategy that strictly increases her payoff. In some settings it would be natural to assume also that a group of individuals will deviate if they have an available joint strategy that strictly increases the payoff of each group member. The *Strong Equilibrium* (SE) concept by Aumann [2] deals with this problem. A profile is a SE if no coalition of agents can jointly deviate in a way that strictly increases the payoff of each coalition member. Intermediate levels of coalitional stability have been suggested, such as stability against deviations of small coalitions (see e.g. [1]), and in particular pairs. An even more appealing solution concept than SE is the *Super-Strong Equilibrium* (SSE) that considers deviations in which no member loses while at least one member makes a positive gain (see, for example, [11]).

A major problem with these proposed solutions is that they seldom exist. Indeed, SSE rarely exist even in cases where strong equilibria do exist (e.g., in simple congestion games [13, 1]), and even if only deviations by pairs are considered.

In this paper we relax the strong requirement that no coalition will have an incentive to deviate, and suggest a quantitative measure to coalitional stability. Assuming we have a Nash equilibrium profile of a game where some pairs of agents can still deviate, we may still wish to measure its stability by referring to the *number* of pairs that have beneficial deviations from that profile. More generally, given a game and a strategy profile, we can associate with it a tuple in which the r-th entry in the tuple is the number of coalitions of size r that can gain by a deviation. This tuple determines the *stability score* of the strategy profile.

Given two strategy profiles, we need a way to decide which one is more stable. A common practice in game theory is to prefer strategy profiles that are *in equilibrium*, i.e. in which there are no unilateral deviations. Since small coalitions are more likely to form and maintain cooperation, a natural extension is to compare stability scores of games with associated strategy profiles using a *lexicographic* ordering of the corresponding vectors.[1] For example, given two n-person games, G_1 and G_2, with respective Nash equilibria s_1 and s_2, the stability score of the former will be higher if the number of beneficial deviations by pairs from s_1 in G_1 is smaller than the number of beneficial deviations by pairs from s_2 in G_2.

While the existence of, say, 19 coalitions that can deviate rather than 15 does not have much significance, we usually care about the behavior in some parameterized family of games where parameters

[1]There are many ways to compare stability score vectors. Choosing the "right" one highly depends on the context and underlying assumptions. However in this paper we avoid such complications by only comparing deviations of coalitions of the same size.

may include number of players, size of the strategy space, etc. If the score of $\mathbf{a}$ is *asymptotically lower* than the score of $\mathbf{a}'$ (w.r.t. one of the parameters), then this may indicate that $\mathbf{a}'$ is substantially more prone to coalitional deviations.

Moreover, when studying such a parametrized family, stability scores may assist us in understanding how the parameters of the game affect coalitional stability. This holds even if there is a unique or a prominent equilibrium.

Stability scores are particularly useful in the context of *mechanism design*, as they allow us to quantify the coalitional stability of various mechanisms and to compare mechanisms that operate in a specific domain. To illustrate this point, we consider two central mechanisms in what is perhaps the most widely studied economic setup in recent years: ad auctions. We analyze in detail the Generalized Second Price (GSP) auction and the Vickrey-Clarke-Groves (VCG) auction, and compare their stability scores.

1.1 Related work

Related solution concepts in games

In the context of non-cooperative games approximate stability is typically measured by the strength of the incentive required to convince an agent to deviate, captured for example by the concept of ϵ-Nash equilibrium. As discussed above, stability against collusion is captured by concepts such as SE and SSE, but these often do not allow a fine distinction between various outcomes.

In addition, coalitions are the key component in *cooperative* game theory, and many variations of coalitional stability have been studied. While we are unaware of solutions concepts that quantify stability by measuring coalitional deviations, models of restricted cooperation capture social constraints that may prevent the formation of some coalitions [18]. Thus a (cooperative) game may not be stable against every coalitional deviation (i.e. have an empty core), but still satisfy all the coalitions that can form in practice. Recently, some papers studied how such social context affects the stability of the game [7, 17]. Moreover, even if some coalition *can* gain by deviation, it may or may not do so: Members of the coalition might intentionally avoid cooperation based on far-sighted prediction (an assumption underlying *coalition-proofness* for example [4]), or just fail to recognize the benefit in deviating. This is especially true if the coalition is large. Stability scores do not assume a particular social context or incentive structure, but simply try and minimize the number of coalitions with profitable deviations.

Collusion and equilibria in ad auctions

Major results of previous work on ad auctions, characterized a special family of equilibria of GSP the auction (used in practice), termed *Symmetric Nash Equilibria*, or SNE (see Section 4.1 for details) [20]. SNEs have many attractive properties which make them a natural choice as outcomes of the GSP auction. Moreover, it has been shown that the SNE leading to the lowest revenue for the seller (termed *Lower Equilibrium* (LE)), coincides with the natural equilibrium of VCG where all bidders report their true values.

The above results led to a surge of papers comparing VCG and the various equilibrium outcomes of GSP, under both public information and private information settings [14, 19, 9, 15]. However, these comparisons focused mainly on revenue, rather than on coalitional stability. The VCG mechanism was shown to be vulnerable to collusion in various domains (see, e.g., [6, 3] for relatively recent work), compared to a simple first-price (pay-your-bid) auction. The formal literature on collusion in second-price auctions goes back to Graham and Marshall [12], while the literature on the more involved matter of collusion in first-price auctions goes back to McAfree and McMillan [16].

1.2 Our contribution

Stability scores are formally defined in Section 2, where we show how they generalize well known solution concepts. In Section 3 we study strict stability scores in a simple family of congestion games. The main purpose of this study is to demonstrate how stability scores can be used in order to compare different Nash equilibria, and to measure how stability is affected by game's parameters. Moreover, while the studied family itself is quite simple, it is often used to model real world situations such as load balancing. Our analysis can give some intuition as to the main factors affecting coalitional stability in such games.

The main results are in Section 4, where we present the VCG and GSP mechanisms for ad auctions (adopting the original model advocated for that setting in the seminal work by Varian [20] and by Edelman et al. [8]), and show bounds on stability scores in these auctions. In particular, we study how the stability of GSP varies as a function of the distributions of agents' valuations and slots' click-through rates, thereby showing that under certain reasonable conditions GSP is far more stable than VCG.

In Section 5 we introduce a modification to the VCG auction that can be used to overcome the observed instability of VCG. In particular, we show that a revised VCG, in which a random reserve price is introduced, induces truth-telling as a super-strong equilibrium.

Omitted proofs are available in the full version of this paper [10].

2. PRELIMINARIES

Games and equilibria

Let $G = \langle N, \{A_i\}_{i\in N}, \{u_i\}_{i\in N}\rangle$ be a normal form game, where $N = \{1, \dots, n\}$ is the set of players, A_i is the set of actions available to player i, and $u_i : \mathbf{A} \to \mathbb{R}$ is player i's utility, where $\mathbf{A} = A_1 \times \cdots \times A_n$ is the set of joint actions (profiles), and for every $\mathbf{a} \in \mathbf{A}$, $u_i(\mathbf{a})$ denotes the utility of player i under action profile $\mathbf{a}$. The vector of actions of all players except player i in the profile $\mathbf{a}$ is denoted by a_{-i}. An action profile $\mathbf{a} \in \mathbf{A}$ is a *Nash Equilibrium* (NE) if $u_i(\mathbf{a}) \geq u_i(b_i, a_{-i})$ for every agent $i \in N$ and every alternative action $b_i \in A_i$.

When considering coalitions, given an action profile $\mathbf{a}$, we denote by a_S the profile of agents in S, and by A_S the set of all such joint actions. The profile of all agents in $N \setminus S$ is denoted by a_{-S}.

Given a profile of actions $\mathbf{a} \in \mathbf{A}$, $b_S \in A_S$ is a *strict deviation* from $\mathbf{a}$ if $u_i(b_S, a_{-S}) > u_i(a_S, a_{-S})$ for every $i \in S$. The profile $\mathbf{a}$ is termed a *Strong Equilibrium* (SE) if there are no $S \subseteq N$ and $b_S \in A_S$, such that b_S is a strict deviation from $\mathbf{a}$.

One can also consider the following weaker notion of deviation. Given a profile of actions $\mathbf{a} \in A$, $b_S \in A_S$ is a *deviation* from $\mathbf{a}$ if $u_i(b_S, a_{-S}) \geq u_i(a_S, a_{-S})$ for every $i \in S$ and there exists $j \in S$ such that $u_j(b_S, a_{-S}) > u_j(a_S, a_{-S})$. The profile $\mathbf{a}$ is termed a *Super-Strong Equilibrium* (SSE) if there is no $S \subseteq N, b_S \in A_S$ that is a deviation from $\mathbf{a}$. Since every strict deviation is clearly a deviation, every SSE is also a SE.

SSE captures the natural requirement that we should resist even situations in which a deviation only benefits some of the deviators without hurting others. A strategy profile is r-SE (respectively, r-SSE) if there are no coalitions of size at most r that have strict deviations (resp., deviations).

Stability scores

The stability score of the profile $\mathbf{a}$ in game G is defined as a vector with n entries. For every $1 \leq r \leq n$, let $\mathcal{D}_r(G, \mathbf{a}) \in \mathbb{N}$ (respectively, $\mathcal{SD}_r(G, \mathbf{a}) \in \mathbb{N}$) be the *number* of coalitions of size r that have deviations (resp., strict deviations) from $\mathbf{a}$ in G. While there are many ways to impose an order on equilibria based on these vectors, we believe that the following lexicographic order is particularly natural.

Given two n-player games G and G' and two profiles $\mathbf{a}$ and $\mathbf{a}'$ in the respective games, we say that the pair $(G, \mathbf{a})$ is *more resistant to deviations* (or *more stable*) than $(G', \mathbf{a}')$, if there exists some $r \leq n$ such that $\mathcal{D}_r(G, \mathbf{a}) < \mathcal{D}_r(G', \mathbf{a}')$ and the terms are equal for every $r' < r$. We can similarly compare strict stability scores to one another.

Our definition of stability score generalizes some widely used notions of stability. For example, $\mathbf{a}$ is a Nash equilibrium (NE) of G iff $\mathcal{D}_1(G, \mathbf{a}) = \mathcal{SD}_1(G, \mathbf{a}) = 0$. This means that the score of a NE (by either definition) is always strictly better than the score of any profile that is not a NE. Further, any profile that is r-SE has a better strict-stability score than any non r-SE profile. A similar property holds w.r.t. r-SSE. As a different example, a profile $\mathbf{a}$ is *Pareto efficient* in G iff $\mathcal{D}_n(G, \mathbf{a}) = 0$.

3. RESOURCE SELECTION GAMES

In this section we demonstrate how stability scores can be used to measure and compare the stability of different outcomes in a given game. To this end we focus on a very simple parametrized family, where games are known to posses at least one pure equilibrium. A natural choice is the family of *resource selection games* (RSG) with identical resources.

In a RSG there is a set of resources $F = \{1, \dots, m\}$, and a non-decreasing cost function $c : [n] \to \mathbb{R}_+$, where $[n] = \{1, \dots, n\}$. Each agent $i \in N$ can select exactly one resource j, and suffers a cost (negative utility) of $c(n_j)$, where n_j is the number of agents that selected resource j. RSGs are *potential games* and thus always admit a pure Nash equilibrium. In fact, any NE $\mathbf{a}$ of a RSG $G = \langle F, N, c \rangle$ is a *strong equilibrium* [13], and thus all equilibria have the same (strict) stability score. However, this is no longer true if the games are concatenated in a sequence.

Formally, a *sequential* RSG (SRSG) is a RSG with k steps. Thus a strategy of an agent $a_i \in F^k$ requires selecting one resource in each step (actions may not depend on the previous steps).[2] We next show that the number of coalitional deviations significantly depends on the played equilibrium. We consider games where $m, n, k \geq 2$, focusing mainly on games with 2 steps.

3.1 Counting deviations: an example

Suppose that $m = 4, n = 6, k = 2$ and that $c(t) = t$ for all $t \leq n$. Any profile in which there are exactly 1 or 2 agents on each resource (in each step) is a Nash equilibrium. However, these equilibria differ in their stability against strict deviation of pairs. Suppose that in the first step agents are partitioned $\{1, 2\}, \{3, 4\}, \{5\}, \{6\}$, and repeat the same actions in the second step. Denote this profile by $\mathbf{a}$. In this case the pair $\{1, 2\}$ can strictly gain as follows: agent 1 joins agent 5 (or 6) in the first step, and agent 2 joins 5 in the second. Thus the cost for each of the two agents drops from 4 to 3. The pair $\{3, 4\}$ can do the same, thus $\mathcal{SD}_2(G, \mathbf{a}) = 2$.

On the other hand, consider a profile $\mathbf{b}$ where players play in the first step as in $\mathbf{a}$, and in the second step are partitioned $\{1, 3\}, \{2, 4\}, \{5\}, \{6\}$; then no pair can strictly gain by deviating. Notice though, that this is still not a strong equilibrium, as the coalition $\{1, 2, 3, 4\}$ can still gain (agents $2, 3$ deviate in the first step, and $1, 4$ in the second), thus $\mathcal{SD}_2(G, \mathbf{b}) = 0$ and $\mathcal{SD}_4(G, \mathbf{b}) = 1$.

Finally, in profile $\mathbf{c}$ agents are partitioned $\{1, 5\}, \{2, 6\}, \{3\}, \{4\}$ (in the second step), and this is a strong equilibrium, i.e. $\mathcal{SD}_r(G, \mathbf{c}) = 0$ for all r. It therefore follows that w.r.t strict stability scores $\mathbf{c}$ is *more stable* than $\mathbf{b}$, which is more stable than $\mathbf{a}$.

Note however that none of these profiles is an SSE or even 2-SSE. More generally, in *any* profile in G there is at least one pair (in fact two) that shares a resource and thus they have a (weak) deviation where just one of them gains. Thus for every profile $\mathbf{p}$ in G, we have that $\mathcal{D}_2(G, \mathbf{p}) \geq 2$.

[2]Equivalently, the game can be described as a routing game, with k sequential parts and m parallel edges in each part.

3.2 Bounding stability scores in two-step RSG

The example above shows that different NE profiles in a particular game may differ in their stability to deviations of pairs or larger coalitions. We want to get a better picture of the gap between the most and least stable NE profiles, focusing on pair deviations. For the results in this section, we will restrict our cost function to be convex.

A nondecreasing cost function $c : [n] \to \mathbb{R}$ is said to be *convex* if it has an increasing marginal loss; i.e., $c(i + 1) - c(i) \leq c(j + 1) - c(j)$ for every $i < j$. Note that when facing a convex cost function, agents in an RSG try to minimize the maximal number of agents using a single resource. If the number of agents on every resource is the same, we say that the partition is *balanced*. If these numbers differ by at most one, we say that the partition is *nearly balanced*.

Let G be a two-step game with a convex cost function. Note that when $n \mod m = 0$, any NE is a balanced partition of agents to resources (in each step). In such partition, no coalition can gain by deviating, as at least one deviating agent will end up paying more in expectation. If, in addition, costs are *strictly* convex, then even weak deviations are impossible. Since in this setting every NE is an SE (and even an SSE), stability scores are trivial. We therefore assume that $n \mod m = q > 0$.

Let $\hat{\mathbf{a}}$ be the profile with the highest number of pair deviations, and let $\mathbf{a}^*$ be the profile with the lowest number of pair deviations.

PROPOSITION 1. $\mathcal{SD}_2(G, \hat{\mathbf{a}}) = \Theta\left(\frac{qn^2}{m^2}\right)$.

PROOF SKETCH OF LOWER BOUND. We note that in $\hat{\mathbf{a}}$ agents play some nearly balanced partition in the first step, and repeat the same partition in the second step. Thus some resources (called *full*) will have $\lceil n/m \rceil$ agents, and the others will have $\lfloor n/m \rfloor$ agents. A crucial observation used in the proof (and in the proofs of the other propositions in this section), is that a pair has a strict deviation if and only if it shares a full resource in both steps. Then (similarly to the example above) one agent switches to a non-full resource in the first step, and the other does the same in the second step. □

Note that when $q = \Theta(m)$, which is a typical situation, there are over $\Omega\left(\frac{n^2}{m}\right)$ deviating pairs.

We find that the best NE $\mathbf{a}^*$ is significantly better than $\hat{\mathbf{a}}$.

PROPOSITION 2. $\mathcal{SD}_2(G, \mathbf{a}^*) = O\left(\frac{n^2}{m^2}\right)$. *Further, if either* $n < m^2$ *or* $q \leq \frac{m}{2}$, *then* $\mathcal{SD}_2(G, \mathbf{a}^*) = 0$, *i.e.* $\mathbf{a}^*$ *is* 2*-SE.*

In order to achieve the upper bound asserted in the proposition we define a profile that tries to scatter in the second step agents that shared a resource in the first step. As a qualitative conclusion, we see that in order to minimize possible deviations, agents should form a partition in the second step that differs as much as possible from the partition in the first step.

3.3 SRSGs with many steps

The following proposition quantifies the stability score of a random pure NE in a RSG with k steps. Note that the set of pure NEs coincides with the set of profiles that are nearly balanced in each step.

PROPOSITION 3. *Let G be an SRSG with k steps and a convex cost function, and let $\mathbf{a}$ be a random NE in G. The expected number of deviating pairs in G is $\mathcal{SD}_2(G,\mathbf{a}) \cong \binom{n}{2}\left(1-(1+\alpha)e^{-\alpha}\right)$, where $\alpha = \frac{q(k-1)}{m^2}$.*

We can summarize how the parameters affect stability as follows. If the number of steps k is small, and the number of resources m increases, then $\alpha \to 0$, and thus $\mathcal{SD}_2(G,\mathbf{a}) \to 0$ as well (i.e. there are very few pairs that can deviate). Conversely, when the number of steps grows (in particular when $k \gg \frac{m^2}{q}$), then almost every pair can deviate with a high probability.

As a corollary of Proposition 3 when $k=2$, we get the lower bound of Proposition 1 for the case $q=\Theta(m)$, as

$$\mathcal{SD}_2(G,\hat{\mathbf{a}}) \geq \binom{n}{2}\left(1-\left(1-\frac{1}{m}\right)\left(1+\frac{1}{m}\right)\right) = \Omega\left(\frac{n^2}{m}\right).$$

4. STABILITY SCORES IN AD AUCTIONS

Having showed how stability scores can be used to analyze coalitional stability in simple games, we next turn to prove our main results. We compute the stability scores of the VCG and GSP ad auctions, which are central to the recent literature on economic mechanism design. Since both auctions admit strong equilibria, we do not consider strict deviations, and instead focus our analysis on weak deviations and the scores they induce.

4.1 Ad auctions: model and notations

An ad auction has s slots to allocate, and $n \geq 2s$ bidders,[3] each with valuation v_i per click [20]. Every slot $1 \leq j \leq s$ is associated with a click-through rate (CTR) $x_j > 0$, where $x_j \geq x_{j+1}$. For mathematical convenience, we define $x_j = 0$ for every $j > s$. Throughout the paper we make the simplifying assumptions that CTRs are strictly decreasing (i.e., $x_j > x_{j+1}$), and that $v_i \neq v_j$ for all $i \neq j$. We denote by bold letter the corresponding vectors of valuations, CTRs, and bids (e.g. $\mathbf{b} = (b_1,\ldots,b_n)$).

A bidder i that has been allocated slot j gains v_i per click (regardless of the slot), and is charged p_j per click. Thus, her total utility is given by $u_i = (v_i - p_j)x_j$.

VCG. In the VCG mechanism every bidder i submits a bid b_i, and the mechanism allocates the j'th slot, $j=1,\ldots,s$, to the j'th highest bidder. Each bidder j is charged (per click) for the "harm" she poses to the other bidders, i.e., the difference between the welfare of bidders $k \neq j$ if j is omitted and their welfare when j exists.

It is well known that the VCG mechanism is *truthful*, meaning that reporting true valuations $b_j = v_j$ is a (weakly) dominant strategy for all bidders. In particular, it is a Nash equilibrium.

Suppose that bidders' valuations are sorted in non-increasing order. Assuming truthful bidding (i.e. $b_j = v_j$ for all j), each bidder $i \leq s$ is allocated slot i, and pays

$$p_i^{VCG} = \sum_{s+1 \geq j \geq i+1} \frac{x_{j-1}-x_j}{x_i} \cdot v_j. \qquad (1)$$

[3]When discussing deviating pairs it is sufficient to assume $n > s$, which is a typical situation. Also, all of our results can be easily adjusted to cases with fewer bidders.

GSP. In the GSP auction, slot j is given to the j'th highest bidder (as in the VCG auction). Denote by j the bidder who is getting slot j. The charge of bidder $j=1,\ldots,s$ equals to the bid of the next bidder; i.e., $p_j = b_{j+1}$. For mathematical convenience, we define $b_{j+1} = 0$ for $j \geq n$.

GSP equilibria. Varian [20] identifies a set of natural Nash equilibria of the GSP auction, termed *envy free NE* or *Symmetric NE* (SNE), which are characterized by a set of recursive inequalities. Varian shows that all SNE's satisfy some very convenient properties. First, in SNE no bidder wants to swap slots with any other bidder.[4] Second, SNEs are efficient in the sense that bidders with higher valuations always bid higher (and thus get better slots). This allows us to assume that valuations are also sorted in non-decreasing order $v_1 \geq v_2 \geq \cdots \geq v_n$. Lastly, SNEs can be easily computed by a recursive formula, which makes them especially attractive for computerized and online settings.

The two equilibria that reside on the boundaries of the SNE set, referred to as *Lower Equilibrium* (LE) and *Upper Equilibrium* (UE), are of particular interest. We denote the LE and UE profiles by $\mathbf{b}^L = (b_i^L)_{i\in N}$ and $\mathbf{b}^U = (b_i^U)_{i\in N}$, respectively. The bids in the LE, for every $2 \leq i \leq s+1$, are given by

$$b_i^L x_{i-1} = v_i(x_{i-1}-x_i) + b_{i+1}^L x_i = \sum_{s+1\geq j\geq i} v_j(x_{j-1}-x_j).$$

In particular, since CTRs are strictly decreasing, we get that $b_i > b_{i+1}$ for all $i \leq s$. A central result by Varian [20] is that the LE equilibrium induces payments, utilities, and revenue equal to those of the truthful outcome in VCG. It is therefore of great interest to compare the stability of these seemingly identical outcomes in both mechanisms.

The bids in the UE, for every $2 \leq i \leq s+1$, are given by

$$b_i^U x_{i-1} = v_{i-1}(x_{i-1}-x_i) + b_{i+1}^U x_i = \sum_{s+1\geq j\geq i} v_{j-1}(x_{j-1}-x_j).$$

In the remaining of this section we measure the stability of the VCG and GSP mechanisms. Our results indicate that while the mechanisms have seemingly identical outcomes, for many natural valuation and CTR functions, GSP is far more stable than VCG.

4.2 Deviations in VCG

Recall that the payment for bidder i is a weighted average of reported (and by truthfulness, the actual) values of bidders $i+1 \leq j \leq s+1$ (see Eq. (1)).

We next characterize the structure of a set of deviators R of size r. We say that a coalition R of r bidders has a *potential to deviate* under VCG (or that it is a *potential coalition*), if either: (a) the group R contains exactly r *winners* (i.e., bidders that are allocated a slot $j \leq s$); or (b) the set R is composed of $t < r$ winners, the first loser, and the $r-t-1$ bidders that directly follow (i.e., bidders $s+1$ through $s+r-t$).

We denote the number of potential coalitions of size r by M_r. We argue that it only makes sense to count potential coalitions when considering a deviation.

To see why, note first that all bidders ranked $s+r$ or worse have no effect on the payment of any other bidder, and can be ignored. Second, the bidders ranked $s+2,\ldots,s+r-1$ are only effective if they allow the bidder allocated slot $s+1$ to lower her bid. Thus non-potential coalitions must contain at least one bidder that has no contribution at all to the deviation, and can therefore be ignored.

[4]When swapping with a bidder in a worse slot, this requirement coincides with the one implied by NE. However when swapping with a bidder in a better slot, envy-freeness is slightly stronger.

Note for example that while adding dummy bidders (with valuation 0) increases the total number of coalitions, the number of potential coalitions remains unchanged.

It is easy to verify that there are $\binom{s}{r}$ coalitions of type (a), and $\sum_{t=1}^{r-1}\binom{s}{t}$ coalitions of type (b). Thus $M_r = \sum_{t=1}^{r}\binom{s}{t}$. Interestingly, in VCG every potential coalition can actually deviate.

PROPOSITION 4. *Under the truthful equilibrium of VCG, denoted by T, any potential coalition has a deviation, i.e., $D_r(VCG, T) = M_r$ for all $2 \le r \le s$.*

PROOF. Let R be some potential coalition, and $i^* \in \operatorname{argmin}_{i \in R} v_i$. We call i^* the *indifferent bidder*. Suppose that every agent $i \in R$ reports v'_i so that $v_i > v'_i > v_{i+1}$. Clearly, this has no effect on slot allocation. In coalitions that include only winners, all the agents except agent i^* (which is indifferent) pay strictly less than their original payments, as the payment monotonically depends on the valuations of the other members of R. In potential coalitions other type, where R includes t winners and $r-t$ losers, all t winners strictly gain. □

4.3 Deviations in GSP

Since LE is a Nash equilibrium, we have that $\mathcal{D}_1(GSP, LE) = \mathcal{SD}_1(GSP, LE) = 0$. In fact, as in the VCG mechanism, no coalition has a strict deviation from the LE profile in GSP. This statement is not as trivial in the GSP mechanism, but it follows from Lemma 9 toward the end of this section. The same analysis holds for the UE in GSP. We next turn to evaluate the resistance of GSP to (non-strict) deviations, focusing on the lower equilibrium. As in the previous section, we only count potential coalitions as all other coalitions necessarily contain redundant participants.

Pair deviations: characterization

We begin by characterizing all deviations by pairs of agents.

Lower equilibrium. It is easy to see that for every $i \le s$, the pair of agents $(i, i+1)$ (called *neighbors*) can always (weakly) gain as a coalition, by having agent $i+1$ lowering her bid to b'_{i+1}, so that $b_{i+1} > b'_{i+1} > b_{i+2}$.[5] In this case, agent $i+1$ is not affected, but agent i gains the difference $x_i(b_{i+1} - b'_{i+1}) > 0$. It is also clear that bidders ranked $s+2$ or worse can never be part of a deviating pair. In terms of the stability score, this means that

$$s \le \mathcal{D}_2(GSP, LE) \le M_2 = \binom{s+1}{2}.$$

Consider the pair of agents (k, j), where $k < j \le s+1$. We want to derive a sufficient and necessary condition under which the pair (k, j) has a deviation. A simple observation is that given some Nash equilibrium, for an agent i to strictly gain by being allocated a new slot $i' \ne i$, the bid $b_{i'+1}$ must strictly decrease, since otherwise this would also be a deviation for i as a single agent (in contradiction to equilibrium). Therefore, either (1) k moves to a worse slot $k' = j-1$, and $b'_j < b_j$; or (2) j moves to a better slot $j' = k$, k is pushed down to $k' = k+1$, and $b'_k < b_k$. However, if j gains in case (2), then this means she is envy in bidder k. This is impossible, as we assumed $\mathbf{b}$ is an SNE. Thus, the only deviation is where $k' = j-1; j' = j$. Further, this is a deviation only if $b_{j-1} > b'_k > b'_j \ge b_{j+1}$. Note that: (i) b'_k can get any value in this range without affecting the utility of k or j, (ii) the utility of j remains the same, and (iii) the most profitable deviation for k is one in which $b'_j = b_{j+1}$ (breaking the tie in favor of j).

[5]The assumption that CTRs are strictly decreasing is required here, as otherwise bidder $i+1$ may not be able to lower her bid.

The discussion above establishes a necessary condition for a pair deviation, and asserts that in every pair deviation of k, j only agent k can strictly gain, where $k < j$. We next complete the characterization by establishing a sufficient condition for pair deviation.

For the following results, we denote $a = x_{j-1} - x_j$ (for our fixed j), and $w_i = \frac{x_{i-1} - x_i}{x_j}$ for all $i \le s+1$.

LEMMA 5. *Suppose that the pair k, j deviates from LE, by moving agent k to slot $k' = j-1$. Let $u(k), u'(k)$ be the utility of agent k before and after the deviation, then*

$$u(k) - u'(k) \ge \sum_{t=k+1}^{j-1}(x_{t-1} - x_t)(v_k - v_t) - a \cdot v_j + a \sum_{i=j+1}^{s+1} w_i v_i.$$

Moreover, in the optimal deviation for agent k the last inequality holds with an equality.

PROOF. Suppose agent j lowers her bid to $b'_j = b_{j+1} + \epsilon$ where $\epsilon \ge 0$ (so j keeps her slot). For any $\mathbf{x}, \mathbf{v}$ the utility of agent k changes as follows:

$$\begin{aligned}
u(k) - u'(k) &= (v_k - b_{k+1})x_k - (v_k - (b_{j+1} + \epsilon))x_{j-1} \\
&= (x_k - x_{j-1})v_k - \sum_{t=k+1}^{s+1}(x_{t-1} - x_t)v_t \\
&\quad + \sum_{i=j+1}^{s+1}\frac{x_{j-1}(x_{i-1} - x_i)}{x_j}v_i + \epsilon x_{j-1} \\
&= \sum_{l=k+1}^{j-1}(x_{l-1} - x_l)v_k - \sum_{t=k+1}^{j}(x_{t-1} - x_t)v_t \\
&\quad + \left(\frac{x_{j-1}}{x_j} - 1\right)\sum_{i=j+1}^{s+1}(x_{i-1} - x_i)v_i + \epsilon x_{j-1} \\
&= \sum_{t=k+1}^{j-1}(x_{t-1} - x_t)(v_k - v_t) - (x_{j-1} - x_j)v_j \\
&\quad + \frac{x_{j-1} - x_j}{x_j}\sum_{i=j+1}^{s+1}(x_{i-1} - x_i)v_i + \epsilon x_{j-1} \\
&= \sum_{t=k+1}^{j-1}(x_{t-1} - x_t)(v_k - v_t) - a \cdot v_j + a\sum_{i=j+1}^{s+1} w_i v_i + \epsilon x_{j-1}.
\end{aligned}$$

The inequality follows since $\epsilon \ge 0$. In the optimal deviation $\epsilon = 0$ in which case we get an equality. Note that $\sum_{i=j+1}^{s+1} w_i v_i$ is a weighted average of valuations. In particular, it is always between v_{s+1} and v_{j+1}. □

As a direct corollary from Lemma 5, we get that in LE the pair k, j (where $k < j-1$), has a deviation *if and only if*

$$\sum_{t=k+1}^{j-1}(x_{t-1} - x_t)(v_k - v_t) < a \cdot v_j - a\sum_{i=j+1}^{s+1} w_i v_i. \quad (2)$$

Upper equilibrium. It is easy to check that a similar characterization to Eq. (2) applies to the UE. However, the conditions differ with respect to bidders that are two positions apart.

PROPOSITION 6. *Given a UE, the pair of agents $i, i+2$ has a deviation for every $i < s$.*

This result holds under all valuation and CTR functions; hence $\mathcal{D}_2(GSP, UE) \ge 2s - 1$. This means that the UE may be slightly

Valuations \ CTR		$\leftarrow$ concave $\rightarrow$	$\leftarrow$ convex $\rightarrow$	
		β-concave	Linear	β-convex
concave	2-concave	All $\binom{s+1}{2}$	All $\binom{s+1}{2}$	-
	Linear	$\Omega(s^2)$	$\Theta(s\sqrt{s})$	$O(s \cdot \log_\beta(s))$
convex	2-convex	-	s	s

Table 1: The table summarizes the number of pairs that have a deviation, i.e., $\mathcal{D}_2(GSP, LE)$. When one function is strictly concave and the other is strictly convex, the score may depend on the exact structure of both functions.

less stable than LE (whose stability is expressed in Theorem 7). Yet, it is not too difficult to show that the number of pair deviations from UE and LE are asymptotically the same. Therefore, in the remainder of this section we focus on stability scores of LE.

Pair deviations: quantification

It turns out that the asymptotic number of pair deviations strongly depends on the shape of both the CTR function and the valuation function. In particular, convexity (as well as concavity and β-convexity) will play a major role in our results. Let $g_1, \dots, g_m$ be a monotonically *nonincreasing* vector.

Similarly to the way defined convex cost functions in Section 3, we say that g is *convex* if it has a decreasing marginal loss; i.e., $g_i - g_{i+1} \geq g_j - g_{j+1}$ for every $i < j$. Similarly, if g has an *increasing* marginal loss then it is *concave*.

Note that linear functions are both convex and concave. A special case of convexity (resp., concavity) is when the marginal loss decreases (resp., increases) exponentially fast.

Let $\beta > 1$. We say that g is *β-convex* if $g_{i-1} - g_i \geq \beta(g_i - g_{i+1})$ for every i. Similarly, g is said to be *β-concave* if $\beta(g_{i-1} - g_i) \leq g_i - g_{i+1}$ for every i. [6]

Intuitively, as either valuations or CTRs are "more" convex,[7] a bidder who deviates by moving to a lower (i.e., worse) slot faces a more significant drop in her utility. Thus we can hope that pairs that are sufficiently distant from one another will not be able to deviate jointly. This intuition is further formalized and quantified in the remainder of this section. For convenience, the results are summarized in Table 1.

The next proposition demonstrates that convexity induces greater stability.

THEOREM 7. *Suppose that both CTR and valuation functions are* convex. *The number of pairs with deviations in the Lower equilibrium can be upper bounded as follows.*

(A) $\mathcal{D}_2(GSP, LE) = O(s\sqrt{s})$.

(B) if CTRs are β-convex then $\mathcal{D}_2(GSP, LE) = O(s \log_\beta s)$.

(C) if valuations are β-convex, for any $\beta \geq 2$, then only neighbor pairs can deviate. I.e., $\mathcal{D}_2(GSP, LE) = s$.

We present the proof of the first statement, so as to demonstrate the proof technique.

PROOF OF 7(A). Recall that $a = x_{j-1} - x_j > 0$. A crucial observation is that $\sum_{i=k+1}^{s+1} w_i v_i$ is in fact a weighted average of valuations, where the weight w_i is proportional to the difference $x_{i-1} - x_i$. Therefore this average is biased toward low values when CTR is convex, and toward high values when it is concave.

Also, since CTRs are convex, we have that for all $i < j$, $x_{i-1} - x_i \geq a$. Thus by Lemma 5,

$$\begin{aligned} u(k) - u'(k) &\geq a \sum_{t=k+1}^{j-1} (v_k - v_t) - a \cdot v_j + a \sum_{i=j+1}^{s+1} w_i v_i \\ &= a \left(\sum_{t=k+1}^{j-1} (v_k - v_t) + \sum_{i=j+1}^{s+1} w_i v_i - v_j \right) \\ &\geq a \left(\sum_{t=k+1}^{j-1} (v_k - v_t) + \operatorname*{avg}_{s+1 \geq i \geq j+1} (v_i) - v_j \right). \qquad (3) \end{aligned}$$

Therefore, in order to prove that the pair j, k can deviate, it is necessary to show

$$\sum_{t=k+1}^{j-1} (v_k - v_t) < v_j - \operatorname*{avg}_{s+1 \geq i \geq j+1} v_i. \qquad (4)$$

We note that under linear CTRs, all inequalities become equalities (in which case Equation (4) is also a sufficient condition). Observe that closer pairs are more likely to deviate. E.g. for pairs s.t. $j = k+2$, it is sufficient that $v_k - v_{k+1} < v_{k+2} - \operatorname*{avg}_{s \geq t' \geq k+3} v_{t'}$ to have a deviation. Let $h = j - 1 - k \geq 1$, and $z = v_k - v_{j-1} = v_k - v_{k+h}$.

From convexity of $\mathbf{v}$ it holds that for all $h' < h$, $\frac{v_k - v_{k+h'}}{h'} \geq \frac{v_k - v_{k+h}}{h} = \frac{z}{h}$, thus for the LHS of Eq. (4),

$$\sum_{t=k+1}^{j-1} (v_k - v_t) \geq \sum_{t=k+1}^{j-1} z \frac{t-k}{h} = \frac{z}{h} \frac{h(h+1)}{2} = \frac{h+1}{2} z. \qquad (5)$$

Bounding the RHS of Eq. (4), we have

$$\begin{aligned} v_j - \operatorname*{avg}_{s+1 \geq i \geq j+1} v_i &\leq v_j - v_{\text{avg}\{s+1 \geq i \geq j+1\}} \qquad \text{(convexity of } \mathbf{v}) \\ &\leq v_j - v_{\left\lceil \frac{j}{2} + \frac{s}{2} \right\rceil} = v_j - v_{\left\lceil j + \frac{s-j}{2} \right\rceil} \\ &\leq \sum_{i'=1}^{\lceil (s-j)/2h \rceil} (v_{j+(i'-1)h} - v_{j+i'h}) \leq \sum_{i'=1}^{\lceil (s-j)/2h \rceil} (v_k - v_{k+h}), \qquad (6) \end{aligned}$$

which is at most $\left\lceil \frac{s-j}{2h} \right\rceil z$. By using the bounds we showed on both sides of the equation, condition (4) implies $h + 1 < \left\lceil \frac{s-j}{h} \right\rceil$, which must be false whenever $h + 1 = j - k > \sqrt{s}$. Therefore each winner $k \leq s$ can deviate with at most $\sqrt{s}$ other bidders, and there can be at most $s\sqrt{s}$ such pairs. □

It is evident from Theorem 7, that convexity can guarantee some level of stability, and further, that "more" convexity can induce more stability. Our next result complements this observation, by showing that *concavity* of valuation and CTR functions affects stability in the opposite direction.

THEOREM 8. *Suppose that both CTR and valuation functions are* concave. *The number of pairs with deviations in the Lower equilibrium can be lower bounded as follows.*

(A) $\mathcal{D}_2(GSP, LE) = \Omega(s\sqrt{s})$.

(B) if CTRs are β-concave for any $\beta > 1$, then $\mathcal{D}_2(GSP, LE) = \Omega(s^2)$ (i.e. a constant fraction of all pairs).

(C) if valuations are β-concave, for any $\beta \geq 2$, then all *pairs can deviate. I.e., $\mathcal{D}_2(GSP, LE) = \binom{s+1}{2} = M_2$.*

[6] Lucier et al. [15] studied GSP auctions with *well-separated* CTR functions, which is a closely related term. In particular, a $\frac{1}{\beta}$-well separated function is also β-convex.

[7] When referring to convexity of CTR/valuation functions, we only consider the first $s + 1$ values.

A linear function is both convex and concave. Therefore, in the special case where both CTRs and valuations are linear, we obtain an asymptotically tight estimation of $\mathcal{D}_2(GSP, LE)$.

Deviations of more than two agents

We first characterize the structure of such deviations.

LEMMA 9. *Suppose that $R \subseteq N$ is a coalition that gains by a deviation, and let b_j, b'_j denote the bids of $j \in R$ before and after the deviation, respectively. Then the following hold:*

(a) There is at least one bidder $i^* \in R$ that does not gain anything from the deviation; this bidder is called the indifferent bidder.

(b) There is at least one bidder $f \in R$ s.t. $R \setminus \{f\}$ still has a deviation; this bidder is called a free rider.

(c) For all $j \in R$, either $b'_j < b_j$, or the utilities of all agents in R (including j) are unaffected by the bid of j.

In order to prove the Lemma, we must show that the bidder that is ranked last among the deviators is an indifferent bidder (i^*). The free rider (f) is either the bidder that is ranked first among the deviators, or some bidder that is isolated of all other deviators. In addition, it is shown that bidders that move to a better slot either strictly lose, or cause some other deviator to strictly lose.

As a direct corollary of Lemma 9, given any coalition R of size ≥ 3, the coalition $R \setminus \{f\}$ can also deviate. By induction, therefore, a coalition R that can deviate always contains a pair that can deviate. Moreover, by part (c) of Lemma 9, it follows that given a deviating coalition of size ≥ 2, it can be extended by adding a bidder who does not change her bid. As a result, a set R can deviate if and only if it contains a pair that can deviate. This crucial observation facilitates the computation of the number of deviations by coalitions of size r for any $r \geq 3$.

Recall that M_r denotes the number of potential coalitions of size r, and that under VCG auction all of these coalitions actually have a deviation. Clearly, $\mathcal{D}_r(GSP, LE) \leq M_r$. We next show how the accurate number of coalitions asymptotically depends on the size of the coalition r and on the number of slots s.

PROPOSITION 10. *If both CTRs and valuations are convex, then*

$$\mathcal{D}_r(GSP, LE) \leq M_r \cdot O\left(\frac{r^2}{\sqrt{s}}\right).$$

In contrast, if both CTRs and valuations are concave, then

$$\mathcal{D}_r(GSP, LE) \geq M_r \cdot d \cdot \left(1 - \exp\left(-\Omega\left(\frac{r\sqrt{r}}{\sqrt{s}}\right)\right)\right)$$

for any positive constant $d < 1$.

That is, at least in the convex case the number of potential deviations under GSP is significantly smaller than under VCG.

This result also establishes an almost sharp threshold for the case of linear CTRs and valuations. In particular, for every $r \gg \sqrt[3]{s}$, almost all coalitions of size r can deviate, while the proportion of coalitions of size $r \ll \sqrt[4]{s}$ that can deviate goes to 0 (when r is fixed and as s grows).

Proposition 10 confirms that the GSP auction is far more stable than the VCG auction against collusions of relatively small coalitions (at least when CTR and valuations are convex).

5. ELIMINATING GROUP DEVIATIONS

5.1 VCG with a reserve price

Consider a variant of the VCG mechanism that adds a fixed reserve price c. That is, only bidders that reports a value of c or higher get a slot, and payments are computed ignoring the other bidders (i.e. replacing their values with c). It is easy to verify that truth-telling remains a dominant strategy, and that Proposition 4 remains valid if the values of all bidders are strictly above c. However, a bidder whose value is exactly c will not join any coalition: by lowering her reported value she will lose her current slot for sure, whereas previously she enjoyed a positive utility.

Now, consider a VCG mechanism that chooses a reserve price as follows. With probability q, the reserve price is chosen randomly from a sufficiently large interval, and with probability $1 - q$, it is set to 0. Crucially, the probability distribution of the reserve price is common knowledge, but agents submit their reports before its realization is revealed. Let us denote the proposed mechanism by VCG*. While the proposed adjustment seems small, it results in a dramatic increase of stability.

THEOREM 11. *If $s \geq n$, then truth-telling is a SSE in VCG*.*

PROOF. First observe that VCG* is a lottery over strategyproof mechanisms, thus no agent has an incentive to deviate unilaterally. Suppose by way of contradiction that there exists a deviating coalition, and let R be such a coalition of minimal size. Since R is minimal, the indifferent agent $i^* \in R$ (as defined in Prop. 4) must lower her reported value, otherwise the coalition $R \setminus \{i^*\}$ can also deviate. Assume, therefore, that $v'_{i^*} = v_{i^*} - \epsilon$ for some $\epsilon > 0$. It is easy to verify that i^* cannot gain in any outcome of the mechanism. In contrast, there is a non-zero probability that c is chosen in the range (v'_{i^*}, v_{i^*}), in which case the utility of i^* becomes 0, compared to $(vi^* - c)x_{i^*} > 0$ under truth-telling. Therefore, agent i^* loses in expectation, contradicting the existence of a coalition R. □

By the last theorem, VCG* guarantees stability whenever $n \leq s$.[8] However, if $s < n$ the bidder ranked $s + 1$ can serve as the indifferent bidder of any coalition. Consequently, VCG* does not posses a SSE. That is, since the utility of agent $s + 1$ is always 0, she will not be discouraged by the random reserve price, even when her reported value falls below the reserve price.

In order to deal with the lack of slots (i.e., the case in which $s \leq n$), we introduce a modified VCG* mechanism, which always induces truth-telling as a SSE.

Consider the following modification to VCG*, termed VCG$^*_\lambda$. Let $0 < \lambda < \frac{1}{n}$. Given some slot $j \leq s$ with a CTR of $x_j > 0$, it is allocated to the bidder that is ranked j with probability $1-\lambda$, and is allocated to the bidder that is ranked $s+1$ with probability λ. This modification effectively creates a new slot $s + 1$, whose expected CTR is λx_j, whereas the new (expected) CTR of slot j becomes $(1 - \lambda)x_j$. This procedure can be applied to the desired additional $n - s$ slots. In particular, a possible instantiation is where the new expected CTR of position s will be $(1 - (n - s)\lambda)x_s$, and there will be $n - s$ new slots with an expected CTR of λx_s. Since the new auction has n slots, the mechanism VCG* can be performed to eliminate all coalitional deviations.

The careful reader will notice that by changing the CTRs, the equilibrium in the new auction may change. However, as long as the order of the slots is preserved, the equilibrium allocation is not affected, and this is ensured by satisfying $\lambda < \frac{1}{n}$. Moreover, the new payment differs from the original payment by at most $v_1 \cdot n \cdot \lambda$; thus for a sufficiently small λ the difference is negligible. As a result, we get the following corollary.

[8]The proof in fact shows a stronger result: truth-telling is a SSE in *dominant strategies*. Thus VCG* is *group-strategyproof.*

COROLLARY 12. *Truth-telling is a SSE in mechanism VCG$^*_\lambda$ for every $0 < \lambda < \frac{1}{n}$. Moreover, the payments and revenue of VCG$^*_\lambda$ can be arbitrarily close to the payments and revenue of VCG.*

5.2 GSP with a reserve price

As evident from the results in the last section, stability of the VCG mechanism is significantly increased by augmenting the mechanism with a random reserve price and additional subtle randomization. It might be tempting to apply the same technique to the GSP mechanism, in an attempt to increase its stability, while maintaining the possibility to achieve a higher revenue than VCG. Unfortunately, this approach fails since (in contrast to VCG) adding a reserve price does not preserve its original set of equilibria.

To see this, consider a GSP mechanism with a fixed reserve price c. Bidder i is affected by the reserve price if either: (I) $v_i > c > b_i$, in which case bidder i has an incentive to raise her bid, as otherwise she will lose the slot; or (II) $v_i < c < b_i$, in which case she has an incentive to lower her bid, as otherwise she will pay more than the slot's worth to her. In both cases it follows that the modified GSP mechanism no longer preserves the SNE properties characterized by Varian (even with respect to unilateral deviations). The reason for the difference between VCG and GSP is that VCG induces truthful revelation in equilibrium; hence cases (I) and (II) suggested above cannot be realized.

6. DISCUSSION AND FUTURE WORK

Our main contribution in this paper is the introduction of *stability scores* — a new stability measure for game equilibria. We demonstrated how stability scores can be used to compare equilibria in congestion games and to draw qualitative results regarding properties of the game and the profiles that increase coalitional stability.

Auctions. Our results indicate that for a prominent class of CTR and valuation functions, GSP is far more stable than VCG.[9]

It is known that the LE of GSP generates exactly the same revenue as VCG, and any other SNE of GSP generates an even higher revenue. This may suggest that GSP is better than VCG with respect to both revenue and stability. However, a relatively simple modification to the VCG mechanism induces a randomized mechanism that eliminates all coalitional deviations, thus turning it into a highly stable mechanism. An open question is whether our results still hold when ads' quality is also considered (see [20]).

Equilibria selection and mechanism design. Analysis of stability scores can be applied to various games and mechanisms. In particular, in games that have multiple Nash equilibria such analysis can aid in selecting the an equilibrium. Understanding how coalitional stability is affected by properties of the game will help us to play better as players, and to create better games as designers.

Toward a realistic picture of coalitional stability. Solution concepts such as ϵ-NE (or ϵ-SE) quantify the benefit an agent or coalition can get from a deviation. Therefore they offer stability under a relaxed notion of self-interest (i.e. agents will only bother to deviate for some substantial gain). In contrast, stability scores still assume purely self-interested agents, but relax a different aspect of coalitional rationality. Practical limitations on information, communication or trust may mean that a coalition of agents will not collude even if they have a potentially high incentive to do so. Other models such as Myerson's [18] assume that limitations on collusion are given in an explicit and structured form.

[9]Empirical studies indicate that CTRs on common platforms are indeed convex, see [5].

In future research we may wish take a combined approach to coalitional stability, considering both known and unknown limitations on collusion, possibly attributing more importance to coalitions with a stronger incentive to deviate. Such models will enable us to better predict realistic outcomes of games, and to improve the mechanisms we design.

7. REFERENCES

[1] N. Andelman, M. Feldman, and Y. Mansour. Strong price of anarchy. In *Proc. of 18th SODA*, pages 189–198, 2007.

[2] R. J. Aumann. Acceptable points in general cooperative n-person games. In *Contribution to the Theory of Games, Vol. IV, Annals of Math. Studies, 40*, pages 287–324. 1959.

[3] Y. Bachrach. Honor among thieves: collusion in multi-unit auctions. In *Proc. of 9th AAMAS*, pages 617–624, 2010.

[4] B. D. Bernheim, B. Peleg, and M. D. Whinston. Coalition-proof Nash equilibria I. Concepts. *Journal of Economic Theory*, 42(1):1–12, 1987.

[5] N. Brooks. The Atlas rank report: How search engine rank impacts traffic. Available from `http://tiny.cc/wj8f3`.

[6] V. Conitzer and T. Sandholm. Failures of the VCG mechanism in combinatorial auctions and exchanges. In *Proc. of 5th AAMAS*, pages 521–528, 2006.

[7] G. Demange. On group stability in hierarchies and networks. *Journal of Political Economy*, 112:754–778, 2004.

[8] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2007.

[9] B. Edelman and M. Schwarz. Optimal auction design and equilibrium selection in sponsored search auctions. *American Economic Review*, 100:597–602, 2010.

[10] M. Feldman, R. Meir, and M. Tennenholtz. Stability scores: Measuring coalitional stability. *CoRR*, abs/1105.5983, 2011.

[11] M. Feldman and M. Tennenholtz. Partition equilibrium. In *Proc. of 2nd SAGT*, pages 48–59, 2009.

[12] D. A. Graham and R. C. Marshall. Collusive bidder behavior at single-object second-price and english auctions. *Journal of Political Economy*, 95:579–599, 1987.

[13] R. Holzman and N. Law-Yone. Strong equilibrium in congestion games. *Games and Economic Behavior*, 21:85–101, 1997.

[14] D. Kuminov and M. Tennenholtz. User modeling in position auctions: re-considering the GSP and VCG mechanisms. In *Proc. of 8th AAMAS*, pages 273–280, 2009.

[15] B. Lucier, R. Paes Leme, and É. Tardos. On revenue in the generalized second price auction. In *The 7th ACM EC Workshop on Ad Auctions*, 2011.

[16] R. P. McAfee and J. McMillan. Auctions and bidding. *Journal of Economic Literature*, 25:699–738, 1987.

[17] R. Meir, J. S. Rosenschein, and E. Malizia. Subsidies, stability, and restricted cooperation in coalitional games. In *Proc. of 22nd IJCAI*, pages 301–306, 2011.

[18] R. B. Myerson. Graphs and cooperation in games. *Mathematics of operations research*, 2(3):225–229, 1977.

[19] D. R. M. Thompson and K. Leyton-Brown. Computational analysis of perfect-information position auctions. In *Proc. of 10th ACM-EC*, 2009.

[20] H. R. Varian. Position auctions. *Int. Journal of Industrial Organization*, 25(6):1163–1178, 2007.

Coalitional Stability in Structured Environments

Georgios Chalkiadakis
Technical University of Crete
Chania, Greece
gehalk@ece.tuc.gr

Evangelos Markakis
Athens University of
Economics and Business
Athens, Greece
markakis@gmail.com

Nicholas R. Jennings
University of Southampton
Southampton, UK
nrj@ecs.soton.ac.uk

ABSTRACT

In many real-world settings, the structure of the environment constrains the formation of coalitions among agents. Therefore, examining the stability of formed coalition structures in such settings is of natural interest. We address this by considering core-stability within various models of cooperative games with structure. First, we focus on characteristic function games defined on graphs that determine feasible coalitions. In particular, a coalition S can emerge only if S is a connected set in the graph. We study the (now modified) core, in which it suffices to check only feasible deviations. Specifically, we investigate core non-emptiness as well as the complexity of computing stable configurations. We then move on to the more general class of (graph-restricted) partition function games, where the value of a coalition depends on which other coalitions are present, and provide the first stability results in this domain. Finally, we propose a "Bayesian" extension of partition function games, in which information regarding the success of a deviation is provided in the form of a probability distribution describing the possible reactions of non-deviating agents, and provide the first core-stability results in this model also.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Economics

Keywords

cooperative game theory, coalition formation, core

1. INTRODUCTION

Cooperative game theory, providing as it does a rich framework for the study of coalition formation among rational players, has in recent years attracted much attention in multiagent systems as a means of forming teams of autonomous agents. The vast majority of work in cooperative game theory assumes that, given a set of agents, any coalition among them is allowed to form. However, in many circumstances the environment imposes restrictions on the formation of coalitions: for reasons that might range from physical limitations and constraints to legal banishments, certain agents might not be allowed to form coalitions with certain others. In many multiagent coordination settings, agents might be restricted to communicate or interact with only a subset of other agents in the environment, due to limited resources or existing physical barriers. In such settings, the environment can be seen to possess some *structure* that forbids the formation of certain coalitions. This can be captured by an undirected graph providing a path connecting any two agents that can belong to the same coalition.

Specifically, sensor networks, communication networks, or transportation networks, within which units are connected through bilateral links, provide natural settings for cooperative games defined over graphs. Another example is provided by hierarchies within an enterprise, where the underlying graph corresponds to a tree.

In this paper we consider various models of cooperative games in *structured* environments as above and study the *stability* of coalition structures in such settings. Stability is one of the key issues in cooperative game settings and examines whether agents have an incentive to depart from an existing coalition structure. As an example, airlines participating in a certain alliance may be willing to move to an alliance that could guarantee them higher profits. Here we focus on the celebrated cooperative stability concept of *the core* [21], which is the set of outcomes that are stable against deviations by any subset of agents. In our work, the definition of the core has to be modified so that the only allowed deviations are those by sets of *connected* coalitions.

Against this background, the rest of the paper is structured as follows. We start with the usual *characteristic function games* (CFGs) setting, under the assumption, however, that the games are defined over graphs. We introduce to the community recent results from the economics literature, which establish the non-emptiness of the core in games defined over a tree, and determine a procedure to find a core element in such games [5]. We then focus on three natural graph structures—lines, trees and cycles—and study the computational problems of (i) deciding the non-emptiness of the core; (ii) finding an element in the core; and (iii) checking if a given outcome belongs to the core. We show certain *positive* results when the underlying graph is a line or a cycle, and when it is a tree and the game is superadditive. These results are interesting from an applications point of view, since many computer or sensor networks exhibit a ring or tree topology. However, for non-superadditive games over trees, certain *negative* complexity results are obtained.

Then, we move on to the more general class of *partition function games (PFGs)* over graphs, and initiate the study of stability in that setting. In PFGs, the value of a coalition depends on the partition currently in place [25]. Defining the core in the presence of *externalities* is complicated and there is no unanimously accepted solution as potential deviators in PFGs have to consider how non-deviators—the "residual" players—would react to their deviation.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

Since residual players can form any structure among themselves, the value of any deviation relies on the resulting partition across the space of agents. A common treatment in the literature is for the deviators to either *pessimistically* assume that non-deviators will partition so as to hurt them the most, or to *optimistically* assume that the partition of the non-deviators will be the best possible [23, 8].We adopt both those views in turn, and provide the first core non-emptiness results in PFG settings with structure. Operating first under the assumption of *pessimism*, we define the (pessimistic) core and then show that for any PFG, there is a corresponding CFG, such that the core of the CFG is contained in the core of the PFG (but the opposite is not always true). Interestingly, this differs from what is known to hold for the class of PFGs where the coalition of all agents is the partition with maximum social welfare (in which case the two cores coincide [8]). This correspondence enables us to generalize the CFG-related results, and show that the core is non-empty for PFGs defined over trees. Furthermore, the same process as before can be used to obtain a core-stable configuration in a PFG defined over a tree, however this will not generally run in polynomial time. We then adopt an *optimistic* view regarding the behaviour of non-deviators, and show that, unlike the pessimistic one, the optimistic core may be empty in PFGs over trees and even over lines. It remains an interesting future work topic to obtain efficient algorithms for special cases of PFGs. As explained in Section 3.1, this seems to be computationally much harder, since it typically involves enumeration over too many partitions. We are not aware of any other work that has tackled stability in PFG settings with structure.

Finally, we propose a natural extension of PFGs, namely *Bayesian partition function games (BPFGs).* In short, instead of resorting to pessimism or optimism, a coalition S of potential deviators in BPFGs assumes that the reaction of the residual players (i.e., what partition they will form if S deviates) is determined by a probability distribution—an assumption that is more realistic and arguably more useful from an AI perspective. We then go on to define the core and initiate its study in this setting as well.

2. CHARACTERISTIC FUNCTION GAMES ON GRAPHS

In this section, we study the issue of stability in the context of characteristic function games defined on graphs. Let $N = \{1, \dots, n\}$ be a set of agents, with $|N| = n$. A subset $C \subseteq N$ is called a coalition. A *characteristic function game (CFG)*—or *coalitional game with transferable utility (TU-game)*—is defined by its function $v : 2^N \mapsto \Re$ that specifies the value $v(C)$ of each coalition C [21]. Intuitively, $v(C)$ represents the maximal payoff the members of C can jointly receive by cooperating, and the agents can distribute this payoff between themselves in any way. A payoff vector $\boldsymbol{x} = \langle x_1, \dots, x_n \rangle$ assigning some payoff to each $i \in N$ is called an *allocation*. We denote $\sum_{i \in C} x_i$ by $x(C)$. Given a partition $\Pi = \{C_1, \dots, C_k\}$, of the agents (we will also refer to a partition as a *coalition structure* interchangeably), an allocation $\boldsymbol{x}$ is called an *imputation* of Π if $x(C_j) = v(C_j)$ for $j = 1, \dots, k$ and $x_i \geq v(\{i\}$ for all i. Note that if $\boldsymbol{x}$ is an imputation for Π, $\sum_{i \in N} x_i = \sum_{C \in \Pi} v(C)$. The set of imputations for Π is $I(\Pi)$.

Assume now that there exists a graph G, which determines the allowed cooperation structures as follows: each node of the graph represents an agent and a coalition C is allowed to form if and only if for every two agents in C there exists a path in the subgraph induced by C that connects them—i.e., the subgraph that is induced by C is a connected subgraph. A *characteristic function game on graph G* is then simply a CFG where v is defined only for coalitions allowed by G. We denote the set of such *feasible coalitions* by $\mathcal{F}(G)$. Similarly, we will refer to *feasible partitions* of feasible coalitions, and we denote the set of all feasible partitions by $P(G)$. Such games can arise naturally in many situations where lack of communication between certain agents makes it impossible for some coalitions to form. Note that when G is a clique then we are back to the usual CFGs where all coalitions are feasible. We also assume that our graph is connected. If not, our findings apply separately to each connected component.

The main stability solution concept in cooperative game theory is, arguably, *the core*—the set of $(\Pi, \boldsymbol{x})$ tuples, where Π is a partition and $\boldsymbol{x}$ an imputation, such that no coalition has an incentive to deviate. However, in our setting it suffices to check only the incentives of the feasible coalitions [17, 4, 5].

DEFINITION 1. *The core of a game with characteristic function $v(\cdot)$ on graph G is the set*

$$C(v, G) = \{(\Pi, \boldsymbol{x}) : \Pi \in P(G), \boldsymbol{x} \in I(\Pi) \wedge x(S) \geq v(S) \, \forall S \, \in \mathcal{F}(G)\}$$

The following observation is straightforward:

FACT 1. *If $(\Pi, \boldsymbol{x}) \in$ core, Π attains maximum social welfare, where the social welfare of Π is: $SW(\Pi) = \sum_{C \in \Pi} v(C)$.*

Though in many games the *grand coalition* of all players might be impossible to form [1], the assumption that it is the one with the highest total welfare—or even the stricter assumption of *superadditivity*, i.e., $v(S \cup T) \geq v(S) + v(T)$ for any disjoint sets S, T—is some times justified. Indeed, the vast majority of work in game theory examines stability in games where the grand coalition emerges. The question of stability then reduces to pairing the grand coalition with an imputation of N.[1]

DEFINITION 2. *When N attains the highest possible social welfare, the core is the set*

$$C(v, G) = \{\boldsymbol{x} \in R^n : x(N) = v(N) \text{ and } x(S) \geq v(S) \, \forall S \in \mathcal{F}(G)\}$$

In most scenarios of interest to multiagent systems, however, it is natural for agents to split into groups to simultaneously perform distinct tasks. Thus, unless explicitly stated, it is *not* required in our games that the grand coalition achieves the highest social welfare.

Network structures have long been recognized as a natural framework for the study of stability. Nevertheless, following the definition of *Myerson value*[2] in [20], research has focused on the question of *building* stable and efficient networks: (mainly pairwise) stability is discussed essentially from a non-cooperative point of view—i.e., w.r.t. *the creation* of stable network structures, through adding or removing links among nodes [14]. In non-cooperative settings, the structural properties of equilibria and the development of algorithms to compute equilibria in *graphical games* which restrict payoff influences among players have also been examined [15].

Networks have also provided inspiration for new representation schemes for coalitional games [13, 3, 2]. Cooperative games with an underlying graph structure have also been considered in the seminal work of [6]. However, they consider games defined on a weighted graph, where the value of a coalition S is the sum of weights of edges that are contained in the subgraph induced by S. Hence, *any* coalition is allowed to form and values are determined by weights, while in our work not all coalitions are feasible and

[1]The core of Definition 1, examining the stability of coalition structures is sometimes referred to as *the CS-core*, while the concept examining the stability of N is referred to simply as *the core* [7].

[2]The Myerson value determines that, while building a network by adding agents one at a time, each agent is assigned his marginal contribution, taking into account all possible orderings of agents.

the characteristic function can be arbitrary and not expressed via weights. Thus, their results are unrelated to ours.

In contrast to the aforementioned approaches, here we care about the stability of *partition-imputation pairs* given a *fixed* underlying network structure that determines the allowed interactions. In particular, we study the complexity of three[3] natural core-related algorithmic questions (Table 1 summarizes our results):

1. **CORE-NONEMPTINESS:** Given a game on a graph G, decide whether $C(v, G) \neq \emptyset$.
2. **CORE-FIND:** Given a game on a graph G, find an element $(\Pi, x) \in C(v, G)$ if $C(v, G) \neq \emptyset$ or output "$C(v, G) = \emptyset$".
3. **CORE-MEMBERSHIP:** Given a game on a graph G and an imputation (Π, x), decide whether $(\Pi, x) \in C(v, G)$.

REMARK 1. *We make the usual assumption that our games are in compact form and are represented by the graph G and an oracle that, for any $S \in \mathcal{F}(\mathcal{G})$, returns the value $v(S)$ in time polynomial in the size of G (e.g. see [9]).*

	Lines (general)	Trees (superadd.)	Trees (general)	Cycles (general)
NONEMPTINESS	O(1)	O(1)	O(1)	P
FIND	P	P	NP-hard	P
MEMBERSHIP	P	co-NP-complete	co-NP-complete	P

Table 1: Core-stability results for CFGs on graphs.

2.1 CORE-NONEMPTINESS and CORE-FIND

We start with the problems of determining whether the core is empty or not and the complexity of finding elements in the core. Related work on solution concepts in graph-restricted games [18, 26] has not addressed issues from an algorithmic point of view, in most cases. The work most relevant to ours is that of Le Breton *et al.* [17], and Demange [4, 5]. In [17] and [4] it is shown that if a game is superadditive and the graph is a tree, the core is non-empty. However, their existential proof does not provide an efficient algorithmic construction of a core element.

Later on, the follow up work of [5] showed that the core is non-empty for trees, even for non-superadditive games. Moreover, Demange proposed a procedure that computes an element in the core. We briefly recall this algorithm below as we will proceed to analyze its complexity, and will also use it in later sections. Originally it was stated in the context of a slightly different model than ours, involving directed graphs, but it is easy to reformulate as:

ALGORITHM 1. *[5] Given a graph G which is a tree, and a characteristic function $v(\cdot)$, first pick a vertex r as the root of the tree. The algorithm consists of two steps.*
Step 1: *Starting from the leaves, compute the* guarantee level *$\hat{g}_i$ of each agent i, inductively: for leaves, $\hat{g}_i$ is simply the reservation value $v(\{i\})$; for an agent that is not a leaf, let $\mathcal{R}^i$ be the set of all subtrees that start at i, i.e., it is the set of all feasible coalitions among i and the agents underneath i. Then: $\hat{g}_i := \max\{v(T) - \sum_{j \in T \setminus \{i\}} \hat{g}_j : T \in \mathcal{R}^i\}$.*
Step 2: *Starting from the root r, pick the coalition T_1 at which $\hat{g}_r$ was attained (breaking ties arbitrarily). Every agent in T_1 receives his guarantee level as a payoff, which is feasible by the definition of $\hat{g}_r$. If $T_1 = N$, we are done. Otherwise, pick a node i not in T_1 whose father belongs to T_1. Pick the coalition from $\mathcal{R}^i$ at which $\hat{g}_i$ was attained, say T_2. All agents in T_2 receive their guarantee level as well. If $T_1 \cup T_2 = N$, we are done, otherwise we continue in the same fashion until we cover N. This produces a partition Π in which all agents receive their guarantee level as their payoff.*

[3]The first problem is included in the second one. However, we feel it is important to study CORE-NONEMPTINESS separately from CORE-FIND because their complexity varies significantly.

THEOREM 1 ([5]). *The outcome produced by the above algorithm belongs to the core.*

To obtain more intuition, it is interesting to observe what the algorithm does in some special cases:

REMARK 2. *If the game is superadditive and the graph is a line from 1 to n, the produced payoff allocation is simply the marginal contribution: $\hat{g}_i = v(\{1, \ldots, i\}) - v(\{1, \ldots, i-1\})$.*

The above theorem implies that CORE-NONEMPTINESS is trivial when the graph is a tree (whether superadditive or not), since the core is always non-empty. Regarding CORE-FIND however, the computational complexity of Algorithm 1 was not addressed in [5]. We therefore now proceed to analyze its complexity. In the usual CFGs, where there is no restriction by a graph, the problem of computing an element in the core, or deciding if the core is non-empty has already been shown to be in co-**NP**, and co-**NP**-complete for certain expressive representation schemes [11]. Here, the fact that the graph may restrict the number of potential deviations gives some hope that the problem may be easier. We show below that this is indeed the case for graphs that are lines or superadditive trees.[4]

THEOREM 2. *For a CFG where (i) the underlying graph is a line or (ii) the game is superadditive and the graph is a tree, CORE-FIND can be solved in polynomial time.*

PROOF. *(i)* For finding an element in the core it is easy to see that Algorithm 1 runs in polynomial time for lines. To prove this, we need to see how many coalitions we need to check when we compute the $\hat{g}_i$ values for each agent i. Suppose *wlog* that the agents are placed in a line starting from agent 1 up to agent n. For agent 1, since it is a leaf, the only coalition we consider is the singleton $\{1\}$. For player 2, to compute $\hat{g}_2$, we need to consider $\{2\}$ and $\{1, 2\}$. Moving on this way we see that for agent $n-1$ we need to consider $\{n-1\}, \{n-2, n-1\}, \ldots, \{1, \ldots, n-1\}$. And finally for agent n, the allowed coalitions are $\{n\}$, $\{n-1, n\}$, $\ldots$,$\{1, \ldots, n\}$. In total for all players we need to consider $1 + 2 + \cdots + (n-1) + n = O(n^2)$ coalitions. Hence it is a polynomial time algorithm.
(ii) Suppose now that the game is superadditive and the graph is a tree. The computational problem that may arise on a tree is the following: if a node i has k children then computing its guarantee level in step 1 of Algorithm 1 requires looking at exponentially many subtrees starting at i, i.e. the set $\mathcal{R}^i$ contains at least 2^k subtrees corresponding to all possible subsets of the children. However, when the game is superadditive, this is not necessary, as implied by the following:

LEMMA 1. *For a node i, let D_i be the tree that starts at i and contains all nodes downwards from i. When the game is superadditive, the guarantee level of every node in step 1 of Algorithm 1 is achieved precisely at D_i.*

The proof of Lemma 1 follows by induction and we omit it here. Given Lemma 1, we can conclude that Algorithm 1 can be implemented in polynomial time in this case. □

[4]Notice, however, that *testing* whether a game is indeed superadditive could still be a hard problem—e.g., it is coNP-complete in the absence of structured environments [10].

Given the results of the above theorem, and Fact 1, it is straightforward to obtain the following corollary:

COROLLARY 1. *For a CFG where the underlying graph is a line, or it is superadditive and the graph is a tree, a partition with maximum social welfare can be found in poly-time.*

When the game is not superadditive, however, CORE-FIND becomes intractable for trees. To see this, we first define:
The social welfare maximization (SW) problem: *Given a game on a graph G, and a rational number k, is there a partition* Π *with* $SW(\Pi) \geq k$?

THEOREM 3. *The* SW *problem when the underlying graph is a tree is* **NP***-complete.*

PROOF. That *SW* is in **NP** is trivial. Just guess a partition and calculate its social welfare. To show that it is **NP**-hard, we reduce from the PARTITION problem, which is the following: given n positive numbers $a_1, \ldots, a_n$ is there a subset S of these numbers such that $\sum_{j \in S} a_j = \sum_{j \notin S} a_j$?

Consider an arbitrary instance of the PARTITION problem with numbers $a_1, \ldots, a_n$. We construct a CFG on a graph which is a tree. In particular, we define a graph with $n+1$ vertices, namely the vertices $\{0, 1, \ldots, n\}$. The only edges are the edges $(0, i)$ for $i = 1, \ldots, n$. Hence the graph is a tree rooted at vertex 0. To determine the game's characteristic function, notice first that feasible coalitions of size at least 2 are forced to contain the root, otherwise they are not connected. Therefore the only allowable coalitions are singletons and sets that contain vertex 0. For the singleton $\{0\}$, set $v(\{0\}) = 0$. For the rest of the singletons, set $v(\{i\}) = a_i$. Finally, the value of coalitions S that contain the root is:

$$v(S) = \begin{cases} \sum_{j \in S \setminus \{0\}} a_j & \text{if } \sum_{j \in S \setminus \{0\}} a_j \neq \sum_{j \notin S} a_j, \\ 1 + \sum_{j \in S \setminus \{0\}} a_j & \text{otherwise} \end{cases}$$

By the definition of $v(S)$, it is easily seen that given $a_1, \ldots, a_n$, a polynomial time oracle for $v(S)$ can be constructed. Set now $k = 1 + \sum_{j=1}^{n} a_j$. This completes the description of the *SW* instance. We can now prove that there exists a set S such that $\sum_{j \in S} a_j = \sum_{j \notin S} a_j$ in the PARTITION problem if and only if there exists a feasible partition with social welfare at least k. Suppose there exists such a set S. Then for the coalition $S \cup \{0\}$ the value is exactly $1 + \sum_{j \in S \setminus \{0\}} a_j$. Hence with the remaining nodes as singletons, we get a partition with social welfare at least k. For the reverse, suppose that there exists a partition with social welfare at least $1 + \sum_j a_j$. If there is no set S that solves the PARTITION problem, then by construction the values of all coalitions are just the sum of the corresponding numbers, and hence the social welfare of any feasible partition is at most $\sum_{j=1}^{n} a_j$, a contradiction. □

Now, we can see why our problem of finding an element in the core is unlikely to have a polynomial time algorithm[5]:

THEOREM 4. *For a general CFG where the underlying graph is a tree, CORE-FIND is* **NP***-hard.*

PROOF. Given Fact 1, any algorithm that finds an element $(\Pi, \boldsymbol{x})$ in the core can solve the *SW* problem, by just calculating the social welfare of the returned Π. □

We now focus on games where the underlying graph is a cycle. These are interesting, as ring topologies are common in networks of many kinds. It can be shown through examples that trees (and forests, if G is disconnected) are the only graphs that guarantee stability irrespective of the function $v(\cdot)$. The presence of cycles can create instances with empty core, even in superadditive games.

THEOREM 5. *For any* $n \geq 3$*, there exist games on n players, where the underlying graph is a cycle and* $C(v, G) = \emptyset$.

For $n = 3$, Theorem 5 follows by the abundance of regular unrestricted CFG's with empty core (since a cycle is a clique for $n = 3$, and hence all coalitions are allowed). For $n \geq 4$ we can construct simple examples of superadditive characteristic functions on cycles, which we omit here.

Hence, the problem CORE-NONEMPTINESS for cycles is not as trivial as in the case of trees. We will show however that we can still have a polynomial time algorithm. We start with superadditive games on cycles. The crucial observation is the following Lemma:

LEMMA 2. *For lines and cycles, the number of feasible coalitions is* $O(n^2)$.

PROOF. A feasible coalition has to be connected and it corresponds to an interval from an agent i to some agent j. This implies a total of $O(n^2)$ since we have at most n choices for i and after fixing i, there can be at most $n-1$ choices for j (multiplied by 2 for cycles, since we then have two paths connecting i and j). □

THEOREM 6. *CORE-NONEMPTINESS and CORE-FIND are in* **P** *for superadditive games on cycles.*

PROOF. From superadditivity, we know that we are looking for an imputation $\boldsymbol{x}$ of the grand coalition. Hence we can check if the following system of linear inequalities has a solution:

$$\sum_{i \in N} x_i = v(N) \text{ and } \sum_{i \in S} x_i \geq v(S) \; \forall S \in \mathcal{F}(G)$$

By Lemma 2, we know that this system has polynomially many constraints and hence can be solved in polynomial time. □

Note that we could have applied the same argument for superadditive games on lines. However, for lines we prefer to use Algorithm 1, since it works for non-superadditive games as well, and it provides a more direct and intuitive way of finding a core element.

One cannot directly use the arguments in Theorem 6 for non-superadditive games on cycles, as an optimal partition is not known *a priori*, and there are exponentially many candidate partitions. However, if one has access to an algorithm that computes an optimal partition Π, then an allocation vector $\boldsymbol{x}$ so that $(\Pi, \boldsymbol{x})$ is in the core, if such an $\boldsymbol{x}$ exists, can be computed via linear programming, by the arguments above. Hence, we need to address the question of whether an optimal partition can be computed in poly-time *over cycles*. Let C be a cycle over n nodes, and let $L_1, ..., L_n$ be the n lines that can be obtained by removing exactly one edge. Then, it is easy to see that the optimal partition of C is either C itself, or the best partition over the optimal partitions of $L_1, ..., L_n$. Thus, one can just run Alg. 1 on $L_1, ..., L_n$ and compare the various solutions with the value of the grand coalition. This proves that CORE-NONEMPTINESS and CORE-FIND are in **P** for general games on cycles. Credit for this proof (provided to us in personal correspondence) goes to G. Greco, E. Malizia, L. Palopoli and F. Scarcello.

Finally, we conclude this subsection with an observation on how to identify more core elements if one has access to algorithms that identify alternative optimal partitions (a result also shown in [10]).

THEOREM 7. *For any game on G, if* $(\Pi, \boldsymbol{x}) \in C(v, G)$*, then* $(\Pi', \boldsymbol{x}) \in C(v, G)$*, for any social welfare maximizing* Π'.

[5]Since our problem is a search problem, it will not belong to **NP**. Our proof essentially shows that our problem is at least as hard as solving an **NP**-complete problem.

2.2 CORE-MEMBERSHIP

In this section we deal with the membership problem. First we show that it can be solved efficiently for lines or cycles.

THEOREM 8. *For a CFG where the underlying graph is a line or a cycle, CORE-MEMBERSHIP can be resolved in* **P**.

PROOF. Given $(\Pi, \boldsymbol{x})$, it is trivial to check if x is an imputation of Π. To check for a successful deviation, we can check all feasible coalitions. By Lemma 2 the proof is complete. $\square$

Concerning trees, the problem is not as easy. We first show that the reduction we used in the proof of Theorem 3 yields a hardness result for general games on trees.

THEOREM 9. *CORE-MEMBERSHIP is co-***NP***-complete for general CFGs on trees.*

PROOF. To check that a $(\Pi, \boldsymbol{x})$ does not belong to the core, it suffices to exhibit a coalition with an incentive to deviate, or check that $\boldsymbol{x}$ is not an imputation of Π. Hence there is a polynomially sized certificate to verify this, which implies that CORE-MEMBERSHIP belongs to co-**NP**(also follows from a result in [10]).

We now show that checking whether $(\Pi, \boldsymbol{x})$ does not belong to the core is **NP**-hard. We use the reduction from the proof of Theorem 3. Given an instance of the PARTITION problem $(a_1, \dots, a_n)$, we construct the game described in Theorem 3 and we consider as a candidate core element the tuple $(\Pi, \boldsymbol{x}) = (N, \langle 0, a_1, \dots, a_n \rangle)$—i.e., we look at the grand coalition with the imputation where the root receives nothing and every other node receives its corresponding number. We claim that $(\Pi, \boldsymbol{x})$ does not belong to the core if and only if there exists a set S that is a solution to the PARTITION problem. To see this, suppose first that there exists a set S such that $\sum_{j \in S} a_j = \sum_{j \notin S} a_j$. Then, by construction and the definition of the characteristic function, the grand coalition is not welfare maximizing, and thus cannot belong to the core (there exists a partition with social welfare at least $1 + \sum_{j=1}^{n} a_j$).

For the other direction suppose that there is no set S for which $\sum_{j \in S} a_j = \sum_{j \notin S} a_j$. This implies that the grand coalition is a welfare maximizing partition and it is also easy to see that every coalition achieves its value $v(S)$ under the imputation $\boldsymbol{x}$. Hence $(\Pi, \boldsymbol{x})$ belongs to the core. $\square$

The reduction above does not imply anything for superadditive trees since it produces non-superadditive instances as well. One could hope that superadditivity makes things easier as was the case for CORE-FIND. The following theorem however reveals that CORE-MEMBERSHIP is hard even for superadditive trees. The proof is based on a simple adaptation of a reduction used for unrestricted games in [10][Theorem 4.1].

THEOREM 10. *CORE-MEMBERSHIP is co-***NP***-complete even for superadditive CFGs on trees.*

PROOF. (**Sketch.**) Checking that the problem is in co-NP is as in Theorem 9. We now give a reduction from SAT. Consider a SAT formula ϕ on n boolean variables $X_1, \dots, X_n$. Let L denote the set of literals, $L = \{X_1, \neg X_1, X_2, \neg X_2, \dots\}$. Given $S \subseteq L$, with $|S| = n$, we say that S is consistent if $\forall X_i \; |S \cap \{X_i, \neg X_i\}| = 1$. For a consistent S, let also $\sigma(S)$ be the truth assignment where all variables that belong to S are set to true and the rest to false.

Given ϕ, we construct a tree with $2n + 1$ nodes. The root is denoted by node 0 and it is connected to $2n$ nodes corresponding to L. This is a star where the only allowable coalitions are either singletons or coalitions containing the root. For any singleton coalition $\{i\}$ we set $v(\{i\}) = 0$. All remaining coalitions are of the form $\{0\} \cup S$ for some $S \subseteq L$. The value of $v(\{0\} \cup S)$ is set to the value of the set S in the proof of [10][Theorem 4.1]. Namely:

$$v(\{0\} \cup S) = \begin{cases} |S|/n, & \text{if } |S| > n, \\ 1 + 1/2n & \text{if } |S| = n, S \text{ is consistent,} \\ & \text{and } \sigma(S) \text{ satisfies } \phi \\ 0, & \text{otherwise} \end{cases}$$

It can be easily proved that this game is superadditive. Furthermore, one can also prove that the allocation where node 0 receives 0 and every other node $1/n$ belongs to the core if and only if the formula ϕ is not satisfiable. We omit the details from this version. $\square$

3. PARTITION FUNCTION GAMES

As mentioned in the introduction, in many circumstances the value of a coalition S does not depend solely on S but is affected by externalities—i.e., $v(S)$ depends on the coalition structure formed by the rest of the agents. To capture such requirements, one is then obliged to move to the more general setting of partition function games. Naturally, therefore, it is of interest to extend the results of Section 2 to *partition function games with structure*. As discussed, the core of PFGs has been studied in economics under specific assumptions. Moreover, recent work has focused on efficient PFG representations and coalition structure generation in PFG settings (Michalak *et al.* [19]; Rahwan *et al.* [22]). However, to date there has been no work on the stability of PFGs defined on graphs.

We begin with some definitions. A PFG is determined by a function $V(\cdot, \cdot)$, specifying the value of a coalition in a certain partition. For $S \in \Pi$, $V(S, \Pi)$ is the value of S when Π forms. In our setting, to define a PFG on a graph G, we further impose that the function $V(\cdot, \cdot)$ is defined only for feasible partitions $\Pi \in P(G)$. We let $V(N)$ denote $V(N, N)$.

Extending the notion of the core to PFGs is not straightforward. This is because, in the presence of externalities, a potential set of deviators S needs to make an assumption on how the rest of the agents behave (i.e., what is the partition that will form in $N \setminus S$) once they deviate. In this section, we focus on the two most common approaches found in the literature, a pessimistic and an optimistic one (see [16] for an overview of PFG solution concepts).

3.1 The Pessimistic Core

We start by defining the core in the case that deviators have a pessimistic view regarding the reaction of the remaining players. For this we need a notion of dominance.

For a given coalition structure $\Pi = (S_1, \dots, S_k) \in P(G)$, a payoff allocation $\boldsymbol{x} = (x_1, \dots, x_n)$ is *feasible* for Π if :

$$\sum_{j \in S_i} x_j \leq V(S_i, \Pi), \quad i = 1, \dots, k$$

Let $\Phi(\Pi)$ denote the set of feasible payoff vectors of a partition Π and let $\Phi = \bigcup_{\Pi \in P(G)} \Phi(\Pi)$. We also define the pessimistic value of a set $S \in \mathcal{F}(G)$ to be $\hat{v}(S) = min_{\Pi \in P(G), S \in \Pi} V(S, \Pi)$.

Consider two payoff vectors $\boldsymbol{x}$ and $\boldsymbol{x}'$ of Φ and a coalition S. We say that ***x′ dominates x via S*** if *(i)* $\sum_{j \in S} x'_j \leq \hat{v}(S)$ and *(ii)* $x'_j > x_j$ for all $j \in S$. The idea behind this type of dominance is that the agents of S are not content with $\boldsymbol{x}$ because there exists another allocation (possibly on a different partition), by which they all get better off without exceeding the total payoff that is guaranteed to S in the worst case partition (which is $\hat{v}(S)$).

We will say that $\boldsymbol{x}'$ pessimistically dominates $\boldsymbol{x}$ and will write $\boldsymbol{x}' dom_{pes} \boldsymbol{x}$ simply if there exists S such that $\boldsymbol{x}'$ dominates $\boldsymbol{x}$ via S. We define the *pessimistic core (p-core)* to be the set of vectors of Φ that are not (pessimistically) dominated by any other member of Φ. This is an extension of the pessimistic core of [8], which was defined for domains without structure and with the grand coalition as the social welfare-maximizing partition.

$$\text{p-core} = \{(\Pi, \boldsymbol{x}) : \Pi \in P(G), \boldsymbol{x} \in \Phi(\Pi), \nexists \boldsymbol{x}' \in \Phi \; s.t. \boldsymbol{x}' \; dom_{pes} \; \boldsymbol{x}\}$$

Given a PFG on a graph G, we will associate with it the following CFG: the set of players and the graph is the same and the characteristic function is $\hat{v}(S)$. The core of this CFG is denoted by:

$$C(\hat{v}, G) = \{(\Pi, \boldsymbol{x}) : \Pi \in P(G), x(S) \geq \hat{v}(S) \forall S \in \mathcal{F}(G) \wedge \boldsymbol{x} \in I(\Pi)\}$$

where $I(\Pi)$ denotes the set of imputations in Π. We now provide a relationship between p-core and $C(\hat{v}, G)$.

THEOREM 11. *For any PFG on a graph G, let $(N, \hat{v}, G)$ be the corresponding (pessimistic) CFG. Then $C(\hat{v}, G) \subseteq$ p-core.*

PROOF. Consider a tuple $(\Pi, \boldsymbol{x})$ that belongs to C$(\hat{v}, G)$. Suppose that $(\Pi, \boldsymbol{x}) \notin$ p-core. Then either $\boldsymbol{x}$ is not feasible for Π or $\boldsymbol{x}$ is dominated by some other feasible vector $\boldsymbol{y}$. But since $(\Pi, \boldsymbol{x}) \in C(\hat{v})$, then $\boldsymbol{x} \in \Phi(\Pi)$, therefore the only possibility is that $\boldsymbol{x}$ is dominated by some other feasible vector. Hence, by definition, there exists a vector $\boldsymbol{y} \in \Phi$ and a set $S \in \mathcal{F}(G)$ s.t. $\boldsymbol{y}$ dominates $\boldsymbol{x}$ via S. That is, $y_j > x_j$ for all $j \in S$ and $\sum_{j \in S} y_j \leq \hat{v}(S)$. Hence $\sum_{j \in S} x_j < \sum_{j \in S} y_j \leq \hat{v}(S)$, which is a contradiction with $(\Pi, \boldsymbol{x}) \in C(\hat{v}, G)$. $\square$

However, the reverse direction is not generally true, as the following example demonstrates:

EXAMPLE 1. *Consider 4 players placed on a line starting from 1 up to agent 4. The idea is to setup the numbers so that the socially optimal partition is $\Pi^* = \{\{12\}, \{34\}\}$, with $V(\{12\}, \Pi^*) = V(\{34\}, \Pi^*) = 10$ and $\hat{v}(\{12\}) < 10$, $\hat{v}(\{34\}) < 10$. Then one can check that $(\Pi^*, \boldsymbol{x})$ with $\boldsymbol{x} = (5, 5, 5, 5)$ belongs to the p-core but not to $C(\hat{v}, G)$. In some detail, $\boldsymbol{x} = (5, 5, 5, 5)$ cannot belong to $C(\hat{v}, G)$, as $\boldsymbol{x}$ is not a valid imputation of partition Π^* in the CFG described by $\hat{v}$, because $\hat{v}(\{12\}) < 10$ and $\hat{v}(\{34\}) < 10$ (in a PFG setting, every S may have a different partition where its pessimistic value $\hat{v}(S)$ is achieved). On the other hand, $\boldsymbol{x}$ is feasible for Π^* in this specific PFG game (and belongs to p-core).*

Note that Theorem 11 does not depend on the structure of the graph and holds for any PFG. This result is interesting for two reasons. First, it demonstrates the differences between domains where the maximum social welfare is achieved by the grand coalition (and by no other partition) and domains where this does not hold. Unlike [8], where they show that, in the former case, the pessimistic core coincides with the core of the CFG, here this is no longer true. Second, Theorem 11, combined with Theorem 1, allows us to establish the non-emptiness of the core in PFGs on trees.

THEOREM 12. *For PFGs where the underlying graph is a tree, the p-core is non-empty.*

For PFGs on trees, the problem of finding an element in the core is **NP**-hard and the membership problem is co-**NP**-complete, since the class of PFGs contains the class of CFGs. Theorem 12 allows us to use Alg. 1 to find an element of the p-core. But even for lines, the algorithm cannot be implemented in polynomial time, unlike CFGs. The problem arises as we need to compute the function $\hat{v}(\cdot)$ to run Alg. 1, because of the exponentially many partitions.

THEOREM 13. *For a PFG on a line and a singleton S, it is* **NP**-*hard to compute $\hat{v}(S)$.*

The same holds whenever $|S| = O(1)$. We omit the proof, which is based on viewing a partition in a line as what is known in combinatorics as an "integer composition" [24]. Algorithmic problems on PFGs are, naturally, computationally more demanding, as the value of a coalition varies across the exponentially many partitions. It would be interesting to identify special classes of PFGs on trees or lines for which a p-core element can be computed in polynomial time. Finally, because of Theorem 5, p-core non-emptiness cannot be guaranteed for PFGs where the underlying graph is a cycle.

3.2 The Optimistic Core

We now consider the opposite approach. Although it is perhaps less intuitive to be optimistic about the reaction of non-deviators, optimism is a well established concept in game theory and economics, as the assumption can be quite natural in certain application domains. For instance, in security games, where players in computer networks attempt to fend off attackers, optimistic players may invest only in self-protection, hoping that the other players will contribute to the overall network protection.[6] As we will see, having optimistic deviators results in higher expectations on their part, and thus the optimistic core may be empty, unlike what was established for the p-core in Theorem 12.

To begin, we need to define a *modified notion of dominance*, which we denote by dom_{opt}. Consider two payoff vectors $\boldsymbol{x}$ and $\boldsymbol{x}'$ of Φ and a coalition S. We say that $\boldsymbol{x}'$ dom_{opt} $\boldsymbol{x}$ (via S) if there exists a partition Π with $S \in \Pi$ such that *(i)* $\sum_{j \in S} x'_j \leq V(S, \Pi)$ and *(ii)* $x'_j > x_j$ for all $j \in S$. The idea here is that since members of S are optimistic, it suffices that they find *some* partition Π and an allocation, feasible for Π, in which they are all better-off.

The *optimistic core (o-core)* is then the set of tuples (Π, x) such that x is not (optimistically) dominated by any other member of Φ.

$$\text{o-core} = \{(\Pi, \boldsymbol{x}) : \Pi \in P(G), \boldsymbol{x} \in \Phi(\Pi), \nexists \boldsymbol{x}' \in \Phi \; s.t. \boldsymbol{x}' \; dom_{opt} \; \boldsymbol{x}\}$$

The following fact is easy to verify:

FACT 2. *o-core $\subseteq$ p-core.*

We will see shortly that the reverse is not true. First we will identify a necessary condition for the o-core to be non-empty. This will be done by establishing a connection of the o-core with the following CFG: the set of players and the graph structure is the same, and, in analogy to Section 3.1, we define now the optimistic value of a set $S \subseteq N$ to be $v^*(S) = max_{\Pi \in P(G), S \in \Pi} V(S, \Pi)$. The resulting CFG is (N, v^*, G), and its core is denoted by:

$$C(v^*, G) = \{(\Pi, \boldsymbol{x}) : \Pi \in P(G), x(S) \geq v^*(S) \forall S \in \mathcal{F}(G) \wedge \boldsymbol{x} \in I(\Pi)\}$$

Then, for a tuple $(\Pi, \boldsymbol{x})$ to belong to the o-core, a necessary condition is that for every S, $\sum_{i \in S} x_i \geq v^*(S)$, as implied by the following theorem.

THEOREM 14. *For any PFG on a graph G, let (N, v^*, G) be the corresponding (optimistic) CFG. Then o-core $\subseteq C(v^*, G)$.*

PROOF. Consider a tuple $(\Pi, \boldsymbol{x})$ that belongs to the o-core. Suppose that $(\Pi, \boldsymbol{x}) \notin C(v^*, G)$. Then by definition, either $\sum_{i \in N} x_i \neq \sum_{S_j \in \Pi} v^*(S_j)$ or there exists a set S such that $\sum_{i \in S} x_i < v^*(S)$. In the first case, since $\boldsymbol{x} \in \Phi(\Pi)$, it follows that $\sum_{i \in N} x_i < \sum_{S_j \in \Pi} v^*(S_j)$, which implies that there exists some set $S_j \in \Pi$ for which $\sum_{i \in S_j} x_i < v^*(S_j)$. Hence we can conclude that in both cases there is some set S for which $\sum_{i \in S} x_i < v^*(S)$. But then we can construct the following vector $\boldsymbol{y}$: Let Π' be the partition where $v^*(S)$ is achieved. We can have a payoff allocation to S so that $\sum_{j \in S} y_j \leq v^*(S)$ and $y_j > x_j \; \forall \; i \in S$. Furthermore, for the remaining players, we can simply allocate payoff to them so as to ensure that for every $S' \in \Pi'$, other than S,

[6]Harikrishna *et al.* [12] have recently introduced a cooperative model against security attacks, but without explicitly examining how the network structure influences agent cooperation decisions.

$\sum_{j \in S'} y_j \leq V(S', \Pi')$. This results in a vector $\boldsymbol{y} \in \Phi(\Pi')$ (and hence $\boldsymbol{y} \in \Phi$) such that $\boldsymbol{y}\ dom_{opt}\ \boldsymbol{x}$ via S, which is a contradiction with the fact that $(\Pi, \boldsymbol{x}) \in$ o-core. $\square$

However, the reverse direction is not generally true, as demonstrated below for games defined on lines.

THEOREM 15. *There exist PFGs, where the underlying graph is a line, for which the o-core is empty.*

PROOF. We consider a graph that forms a line from agent 1 to agent 4 (in a similar manner one can generalize the example to lines with a larger number of agents). This gives rise to 8 feasible partitions, with corresponding coalitional values as follows.

Partition	Value	Value of coalitions
$\{1\},\{2\},\{3\},\{4\}$	12	$(3,3,3,3)$
$\{1\},\{2\},\{3,4\}$	11	$(2,2,7)$
$\{1\},\{2,3\},\{4\}$	11	$(2,7,2)$
$\{1\},\{2,3,4\}$	11	$(2,9)$
$\{1,2\},\{3\},\{4\}$	19	$(11,4,4)$
$\{1,2\},\{3,4\}$	20	$(10,10)$
$\{1,2,3\},\{4\}$	18	$(15,3)$
$\{1,2,3,4\}$	19	19

We can show that for any of these feasible partitions, there can be no vector $\boldsymbol{x}$ with $\boldsymbol{x} \in \Phi(\Pi)$ such that $(\Pi, \boldsymbol{x}) \in$ o-core. To see this, consider for example the welfare maximizing partition $\Pi = (\{1,2\},\{3,4\})$. For any $\boldsymbol{x} \in \Phi(\Pi)$, we know that $x_1 + x_2 \leq 10$. But then we can construct a vector $\boldsymbol{y}$, feasible for the partition $(\{1,2\},\{3\},\{4\})$, in which $y_1 + y_2 \leq 11$ and $y_1 > x_1, y_2 > x_2$. This means that $\boldsymbol{y}\ dom_{opt}\ \boldsymbol{x}$ and $(\Pi, \boldsymbol{x})$ cannot belong to the o-core. If we consider the partition $\Pi' = (\{1,2\},\{3\},\{4\})$, then we know that for any $\boldsymbol{x} \in \Phi(\Pi')$, $x_3 + x_4 \leq 8$. But then we can construct in a similar manner a vector $\boldsymbol{y}$ that is feasible for Π and dominates $\boldsymbol{x}$ via the set $\{3,4\}$. By using similar arguments, we can also conclude that for the remaining 6 partitions, we cannot find a vector that would be feasible for them and undominated. $\square$

On the contrary, $C(v^*, G)$ is non-empty on trees by Theorem 1. This shows that unlike the p-core, which is guaranteed to exist in PFGs over trees, the o-core is a stricter concept and cannot exist for all such games. Intuitively, this is because optimistic expectations lead to a larger set of potential deviations and eventually to the absence of stable configurations.

4. BAYESIAN PARTITION FUNCTION GAMES

In this section, we initiate the study of a model that is even more general than the usual partition function games. As mentioned, the PFG literature has mostly focused on scenarios where deviators are assumed to be either pessimistic or optimistic with respect to others' behaviour. Even in approaches that attempt to move away from these extremes, definitions of dominance rely on some degree of optimism or pessimism—as, e.g., in the *recursive core* model of [16]. In many realistic scenarios, however, information regarding the behaviour of the residual agents, when a set S decides to work on its own, might be available to the deviators in the form of a probability distribution. Such a distribution could be derived from market data available, observation of historical evidence and trends, domain knowledge and so on. Arguably, it is far more useful to multiagent systems research and practice to assume that agent uncertainty is described by a probability distribution, rather than restrict attention to just one or two possible scenarios. Apart from better reflecting real life situations, such an extension could potentially allow for Bayesian inference and learning in PFGs; something that has not been discussed in the PFG literature.

We thus proceed to define *Bayesian partition function games* as normal PFGs equipped with probability distributions specifying the likelihood of a partition emerging when a specific coalition forms.

DEFINITION 3. *A* Bayesian partition function game (BPFG) *is a tuple $\mathcal{B} = \langle \mathcal{P}, Pr_S(\cdot) \rangle$, where $\mathcal{P}$ is a PFG and for every S, $Pr_S(\cdot)$ is a probability mass function, specifying the probability $Pr_S(\Pi)$ that $\Pi \in P(G)$ emerges given that $S \in \Pi$ forms.*

For each S, it should hold that $\sum_{\Pi \ni S} Pr_S(\Pi) = 1$. Also for the grand coalition, it holds that $Pr_N(\{N\}) = 1$. For a set S, the probabilities $Pr_S(\cdot)$ reflect the beliefs of S on the reaction of the residual agents, when S is considering to deviate. Then, the *expected value* that a coalition S would receive in a potential deviation in a BPFG is: $\tilde{v}(S) = E_\Pi[V(S,\Pi)] = \sum_{\Pi \ni S} Pr_S(\Pi) V(S,\Pi)$.

It is now reasonably straightforward to define the Bayesian PFG-core as the set of *partition-allocation pairs* with efficient allocations that are weakly preferable in expectation to any potential deviation by some coalition S.

DEFINITION 4 (BPFG-CORE). *The* BPFG-core *is the set of $(\Pi, \boldsymbol{x})$ pairs where $\sum_{i \in C} x_i = V(C, \Pi), \forall C \in \Pi$* and for any feasible coalition, *$S \in \mathcal{F}(G)$, it holds that $x(S) \geq \tilde{v}(S)$.*

We note here that, interestingly, even if $(\Pi, \boldsymbol{x})$ belongs to the BPFG-core, Π may not necessarily be a welfare maximizing partition. This is because coalitions only judge whether the payoff they receive is good in expectation—i.e., at least $\tilde{v}(S)$. If an optimal partition exists that is better than Π but occurs only with a small probability, then this does not necessarily prevent the stability of Π. Though this is a departure from the usual models where the core is defined, it is very natural in a Bayesian setting.

Finding necessary and sufficient conditions for characterizing the elements of the BPFG-core is naturally of interest. Here, we provide one *sufficient* condition for a $(\Pi, \boldsymbol{x})$ element to be in the BPFG-core. First, consider the maximum attainable value $v^*(S)$ of a feasible coalition S (i.e., its best possible value under any potential partition containing S). Then, the following fact holds:

FACT 3. $v^*(S) \geq \tilde{v}(S)\ \forall\ S \in \mathcal{F}(G)$.

By Fact 3, we have:

THEOREM 16. *Let $(\Pi, \boldsymbol{x})$ be a Bayesian partition function game outcome with $\sum_{i \in C} x_i = V(C, \Pi), \forall C \in \Pi$. $(\Pi, \boldsymbol{x})$ is in the BPFG-core if the following condition holds:*

$$x(S) \geq v^*(S), \forall S \in \mathcal{F}(G)$$

Note that this a sufficient condition for an element to be in the BPFG-core, without the need to take into account probability distribution estimates, even though the setting is probabilistic. Hence, if there is information available about the maximum values $v^*(S)$, then one may not need to go through computing all expectations.

This however is not a *necessary* condition: it may occur that an element $(\Pi, \boldsymbol{x})$ is in the core and there is an S with $v^*(S) > \sum_{i \in S} x_i$. This is demonstrated in the following example.

EXAMPLE 2. *Consider $N = \{1,2,3,4\}$, and suppose $V(N) = 120$ and $\boldsymbol{x} = \{30,30,30,30\}$. We will argue about the tuple $(N, \boldsymbol{x})$. Consider the coalition $S = \{1,2\}$ of agents considering to deviate, with beliefs specifying that for the two possible partitions to emerge if S breaks away, say Π_1, Π_2, $Pr_S(\Pi_1) = Pr_S(\Pi_2) = 0.5$. Let $V(S, \Pi_1) = 100$, $V(S, \Pi_2) = 10$. Let also the value of any other feasible coalition in this game be zero in any partition. Then, since $\tilde{v}(S = \{1,2\}) = 55$ while $x(S) = 60$, and given that all non-mentioned coalitions have zero value, it holds that for any feasible C in this setting, $x(C) \geq \tilde{v}(C)$, therefore $(N, \boldsymbol{x})$ is in the BPFG-core. However, for the given S, $v^*(S) = 100 > x(S)$.*

In analogy to the definition of the core in superadditive CFGs, we can also define here the core with respect to the grand coalition (*BPFG-core-grand*), as the set of efficient allocations dividing up the value of N that make the deviation of a set of agents unprofitable in expectation.

DEFINITION 5 (BPFG-CORE-GRAND). *The* BPFG-core-grand *is the set* $\boldsymbol{x}$ *of allocations such that* $x(N) = V(N)$, *and* $\forall S \in \mathcal{F}(G)$, $x(S) \geq \tilde{v}(S)$.

Since $\tilde{v}(N) = V(N)$, it is easy to verify from Def. 2 and Def. 5 that the BPFG-core-grand and the core of the CFG with characteristic function $\tilde{v}(\cdot)$ coincide.

FACT 4. *Let $\mathcal{B}$ be a BPFG. Consider the CFG defined by the function $\tilde{v}(\cdot)$. Then,* BPFG-core-grand$(\mathcal{B}) = C(\tilde{v}, G)$.

Using Fact 4 and Theorem 1 we can now establish:

THEOREM 17. *If $\mathcal{B}$ is a BPFG defined on a tree G, then its BPFG-core-grand is non-empty.*

The discussion right after Theorem 12, including Theorem 13, applies for $\tilde{v}(\cdot)$ as well. Hence, even though one could use Algorithm 1 to find an element of BPFG-core-grand, this cannot be done in polynomial time. It would be interesting to identify special cases of BPFGs that admit polynomial time algorithms.

We believe that BPFGs are a natural setting that deserves further exploration. Clearly, it would be interesting to obtain a correspondence between the CFG core with coalition structures and the general BPFG-core. However Theorem 17 does not hold for the general BPFG-core because the analog of Fact 4 is not always true. Namely, an element $(\Pi, \boldsymbol{x}) \in C(\tilde{v}, G)$ satisfies $x(C) = \tilde{v}(C)$ for every $C \in \Pi$. But this may not be a valid allocation for the BPFG since it may not hold that $x(C) = V(C, \Pi)$ (i.e., the feasibility of an allocation $\boldsymbol{x}$ would have to be assessed w.r.t. Π). Hence the task of mapping the CFG core to the BPFG-core is considerably more challenging when coalition structures are involved.

5. CONCLUSIONS AND FUTURE WORK

In this paper we studied core-stability in several models of cooperative games defined on graphs that constrain the formation of coalitions. First, we obtained complexity results in the usual CFG setting, several of which are *positive* for certain graph structures of interest, such as trees and cycles which form the backbone of networks found in the real world. We then initiated the study of core-stability in PFGs defined over graphs, examining it both from a pessimistic and an optimistic viewpoint. Furthermore, we proposed a Bayesian model for PFGs, which we believe is more realistic than the usual models in economics, and suits better the coalition formation paradigms of interest to multiagent systems. We took some steps towards the study of the core in this model as well.

Regarding future work, we are particularly interested in exploring the PFG and Bayesian PFG domains further, as outlined above. We also envisage linking theoretical results in these domains to real-world applications. For instance, tractable algorithms to identify ϵ-stable coalitions could be used to inform planning decisions and optimize task execution in structured multiagent settings.

Acknowledgments

We are grateful to G. Greco, E. Malizia, L. Palopoli and F. Scarcello for extremely helpful discussions. Georgios Chalkiadakis was partially supported by the EU FP7-ICT Cognitive Systems, Interaction, and Robotics under the contract #270180 (NOPTILUS).

6. REFERENCES

[1] R. J. Aumann and J. H. Dreze. Cooperative Games with Coalition Structures. *Intern. Journal of Game Theory*, 3(4):217–237, 1974.

[2] Y. Bachrach, R. Meir, K. Jung, and P. Kohli. Coalitional Structure Generation in Skill Games. In *AAAI-2010*.

[3] R. I. Brafman, C. Domshlak, Y. Engel, and M. Tennenholtz. Transferable utility planning games. In *AAAI-2010*.

[4] G. Demange. Intermediate Preferences and Stable Coalition Structures. *J. of Mathematical Economics*, 23:45–58, 1994.

[5] G. Demange. On Group Stability in Hierarchies and Networks. *Journal of Political Economy*, 112(4):754–778, August 2004.

[6] X. Deng and C. Papadimitriou. On the complexity of cooperative solution concepts. *Math. of Operation Research*, 19:257–266, 1994.

[7] E. Elkind, G. Chalkiadakis, and N. R. Jennings. Coalition structures in weighted voting games. In *ECAI-2008*, pages 393 – 397, 2008.

[8] Y. Funaki and T. Yamato. The core of an economy with a common pool resource: A partition function form approach. *International Journal of Game Theory*, 28(2):157–171, 1999.

[9] G. Greco, E. Malizia, L. Palopoli, and F. Scarcello. On the Complexity of Compact Coalitional Games. In *IJCAI-2009*.

[10] G. Greco, E. Malizia, L. Palopoli, and F. Scarcello. On the complexity of the core over coalition structures. In *IJCAI-2011*.

[11] G. Greco, E. Malizia, L. Palopoli, and F. Scarcello. On the complexity of core, kernel, and bargaining set. *Artificial Intelligence*, 175(12-13):1877–1910, 2011.

[12] N. Harikrishna, V. Venkatanathan, and C. Pandu Rangan. Towards a cooperative defense model against network security attacks. In *Workshop on the Economics of Information Security (WEIS)*, 2010.

[13] S. Ieong and Y. Shoham. Marginal Contribution Nets: A Compact Representation Scheme for Coalitional Games. In *Proc. of ACM-EC'05*, pages 193 – 202, 2005.

[14] M. Jackson. A Survey of Models of Network Formation: Stability and Efficiency. In G. Demange and M. Wooders, editors, *Group Formation in Economics; Networks, Clubs and Coalitions*, 2004.

[15] S. Kakade, M. Kearns, J. Langford, and L. Ortiz. Correlated Equilibria in Graphical Games. In *ACM-EC'03*, pages 42 – 47, 2003.

[16] L. Koczy. A recursive core for partition function form games. *Theory and Decision*, 63(1):41–51, August 2007.

[17] M. Le Breton, G. Owen, and S. Weber. Strongly Balanced Cooperative Games. *Int. Journal of Game Theory*, 20:419–427, 1992.

[18] R. Meir, J. S. Rosenschein, and E. Malizia. Subsidies, stability, and restricted cooperation in coalitional games. In *IJCAI-2011*, pages 301–306, Barcelona, 2011.

[19] T. Michalak, D. Marciniak, M. Szamotulski, T. Rahwan, M. Wooldridge, P. McBurney, and N. R. Jennings. A logic-based representation for coalitional games with externalities. In *AAMAS-2010*, pages 125–132, 2010.

[20] R. Myerson. Graphs and Cooperation in Games. *Mathematics of Operations Research*, 2(3):225–229, 1977.

[21] R. Myerson. *Game Theory: Analysis of Conflict*. 1991.

[22] T. Rahwan, T. Michalak, N. R. Jennings, M. Wooldridge, and P. McBurney. Coalition Structure Generation in Multi-Agent Systems With Positive and Negative Externalities. In *IJCAI-09*, 2009.

[23] P. P. Shenoy. On coalition formation: A game-theoretical approach. *International Journal of Game Theory*, 8(3):133–164, 1979.

[24] R. Stanley. *Enumerative Combinatorics (Vol. 1)*. Cambridge University Press, 1997.

[25] R. Thrall and W. Lucas. n-Person Games in Partition Function Form. *Naval Research Logistics Quarterly*, 10(1):281–298, 1963.

[26] R. van den Brink. On Hierarchies and Communication. Tinbergen Institute and Free University Discussion Paper 06/056-1, 2006.

Overlapping Coalition Formation Games: Charting the Tractability Frontier

Yair Zick
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
yair0001@ntu.edu.sg

Georgios Chalkiadakis
Department of Electronic and Computer Engineering
Technical University of Crete, Greece
gehalk@ece.tuc.gr

Edith Elkind
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
eelkind@ntu.edu.sg

ABSTRACT

Cooperative games with overlapping coalitions (OCF games) [3, 23] model scenarios where agents can distribute their resources among several tasks; each task generates a profit which may be freely divided among the agents participating in the task. The goal of this work is to initiate a systematic investigation of algorithmic aspects of OCF games. We propose a discretized model of overlapping coalition formation, where each agent $i \in N$ has a weight $w_i \in \mathbb{N}$ and may allocate an integer amount of weight to any task. Within this framework, we focus on the computation of outcomes that are socially optimal and/or stable. We discover that the algorithmic complexity of the associated problems crucially depends on the amount of resources that each agent possesses, the maximum coalition size, and the pattern of interaction among the agents. We identify several constraints that lead to tractable subclasses of OCF games, and provide efficient algorithms for games that belong to these subclasses. We supplement our tractability results by hardness proofs, which clarify the role of our constraints.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Theory

Keywords

Overlapping Coalition Formation, Complexity, Bounded Treewidth

1. INTRODUCTION

In many multiagent systems, agents split into teams in order to complete tasks or solve problems [14]. Typically, the collective efforts of a team are rewarded with a payoff, which then needs to be shared among the team members. When agents are selfish, i.e., aim to maximize their own payoff, such settings are modeled using the tools of *cooperative game theory*, which suggests a variety of approaches to team formation and payoff division [2].

The usual framework of cooperative games assumes that agents are divided into disjoint sets in order to perform tasks. However, sometimes the agents may benefit from splitting their resources among several jobs and forming *overlapping* coalitions. The study of such scenarios was initiated by Shehory and Kraus [21] (see also Dang et al. [6]), who assumed that agents are fully cooperative. More recently, Chalkiadakis et al. [3] proposed a formal model for cooperative games with overlapping coalition structures (OCF games), which, in contrast to the previous work, takes agent incentives into account. More specifically, Chalkiadakis et al. [3] define games in which agents can form *partial coalitions*, and an agent can participate in multiple (overlapping) coalitions. Such coalitions correspond to vectors in $[0, 1]^n$: the i-th coordinate of the vector identifies the *fraction* of the i-th agent's resources devoted to this coalition.

The main focus of [3] is coalitional stability in OCF games. In such games, identifying outcomes that are stable, i.e., resistant to group deviations, is more difficult than in the standard model. This is because in OCF games one needs to take into account the non-deviators' reaction to deviation. Indeed, when no overlapping coalitions are allowed, the deviators do not care about the reaction of other agents to their actions: all of their resources are now devoted to maximizing their own welfare and they have no stake in what other agents do. In contrast, when agents are allowed to divide their resources among several tasks, group reaction to deviation must be taken into consideration: the deviating agents may remain involved in one or more partial coalitions with non-deviators, and they need to reason about the payoff they expect to get from such collaborations. These issues were raised in [3] and subsequently studied in detail by Zick and Elkind [23], who proposed a general framework for handling coalitional reaction to group deviations under a number of solution concepts.

While Chalkiadakis et al. [3] and Zick and Elkind [23] provide a rich framework for reasoning about OCF games, they give short shrift to computational aspects of such games. Indeed, Zick and Elkind [23] ignore the algorithmic efficiency issues altogether, and in [3] the algorithmic results are limited to a special class of OCF games known as *Threshold Task Games*; while these games supply a useful testing ground for comparing different stability concepts, they are clearly not expressive enough to capture all OCF games. We remark that, in general, designing efficient algorithms for coalitional games, even in the absence of overlapping coalitions, is a challenging task. Indeed, the sheer number of such games precludes the existence of a representation scheme that can describe any coalitional game in $\text{poly}(n)$ bits (where n is the number of players). For this reason, a number of different representation languages for coalitional games have been proposed, with each language capturing a specific family of application scenarios and requiring purpose-built algorithms for computing various solution concepts (see, e.g., Chalkiadakis et al. [4] for a literature review).

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

For games with overlapping coalitions, the situation is even more dire: even identifying a finitary representation for OCF games is non-trivial, as we need to be able to specify the payoff of each partial coalition, and there are infinitely many such coalitions.

The aim of this paper is to initiate a systematic investigation of algorithmic aspects of OCF games. We propose a discretized model of overlapping coalition formation, where each agent $i \in N$ has a weight $w_i \in \mathbb{N}$ and may allocate an integer amount of resources to any partial coalition. This simplification ensures the existence of a finitary representation, and can be justified by observing that in practice, agents' resources have certain granularity and therefore cannot be divided with arbitrary precision. We then focus on the computation of outcomes that are socially optimal and/or stable. We discover that the complexity of the associated problems crucially depends on the amount of resources that each agent possesses, the maximum coalition size, the pattern of interaction among the agents, and the properties of the arbitration function. We identify the constraints that lead to tractable subclasses of OCF games, and provide efficient algorithms for games that belong to these subclasses. We supplement our tractability results by hardness proofs, showing that the constraints that we impose are, in a sense, necessary. Our results suggest a number of future research directions; we hope that they will serve as a starting point for a comprehensive algorithmic analysis of OCF games, which is a necessary precondition for the practical applicability of such games.

2. PRELIMINARIES

Throughout the paper, we use boldface lowercase letters to denote vectors and uppercase letters to denote sets. Given two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we write $\mathbf{x} \leq \mathbf{y}$ when $x_i \leq y_i$ for all $i = 1, \ldots, n$.

Cooperative Games with Overlapping Coalitions We briefly describe the model as presented in [3]. A *cooperative game with overlapping coalitions*, also referred to as an *overlapping coalition formation (OCF) game*, is given by a set of *agents* $N = \{1, \ldots, n\}$ and a *characteristic function* $v : [0,1]^n \to \mathbb{R}_+$; we write $G = (N, v)$. Each agent has one unit of resource (time, money, etc.); using their resources, agents may form *partial coalitions*: a partial coalition is described by a vector $\mathbf{c} = (c_1, \ldots, c_n)$, where c_i is the fraction of i's resource dedicated to this coalition. The *value* of the partial coalition $\mathbf{c}$ is given by $v(\mathbf{c})$, and its *support* is given by $\mathit{supp}(\mathbf{c}) = \{i \in N \mid c_i > 0\}$. A *coalition structure* over a subset S of N is a collection $CS = (\mathbf{c}^1, \ldots, \mathbf{c}^m)$ of partial coalitions that satisfies $\sum_{j=1}^m \mathbf{c}^j \leq \mathbf{e}^S$, where $\mathbf{e}^S \in \{0,1\}^n$ is the indicator vector of the set $S \subseteq N$. We denote by $\mathcal{CS}(S)$ the set of all coalition structures over S. Given a coalition structure $CS = (\mathbf{c}^1, \ldots, \mathbf{c}^m)$, we overload notation and write $v(CS) = \sum_{j=1}^m v(\mathbf{c}^j)$; we refer to $v(CS)$ as the *value* of CS. The *superadditive cover* of a characteristic function v is the mapping

$$v^*(\mathbf{c}) = \sup_{CS \in \mathcal{CS}(N)} \{v(CS) \mid \sum_{\mathbf{c}' \in CS} \mathbf{c}' \leq \mathbf{c}\},$$

which computes the most that the agents can earn if their resources are given by $\mathbf{c}$.

The payoff $v(\mathbf{c}^j)$ needs to be divided among agents who contribute to $\mathbf{c}^j$, i.e., members of $\mathit{supp}(\mathbf{c}^j)$; a *division of payoffs* of a partial coalition $\mathbf{c}^j$ is a vector $\mathbf{x}^j \in \mathbb{R}_+^n$ that satisfies $\sum_{i=1}^n x_i^j = v(\mathbf{c}^j)$, $x_i^j = 0$ for any $i \notin \mathit{supp}(\mathbf{c}^j)$. A *pre-imputation* for a coalition structure $CS = (\mathbf{c}^1, \ldots, \mathbf{c}^m)$ is a collection of vectors $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^m)$, where for each $j = 1, \ldots, m$ the vector $\mathbf{x}^j$ is a division of payoffs of coalition $\mathbf{c}^j$. The set of all pre-imputations for a coalition structure CS is denoted by $\mathcal{I}(CS)$. The pair $(CS, \mathbf{x})$, where $\mathbf{x} \in \mathcal{I}(CS)$, is called a *feasible outcome*. The *total payoff* of a player i under a feasible outcome $(CS, \mathbf{x})$ with coalition structure $CS = (\mathbf{c}^1, \ldots, \mathbf{c}^m)$ and a payoff vector $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^m)$ is defined as $p_i(CS, \mathbf{x}) = \sum_{j=1}^m x_i^j$; we extend this notation to sets of agents by setting $p_S(CS, \mathbf{x}) = \sum_{i \in S} p_i(CS, \mathbf{x})$ for $S \subseteq N$.

Stability and Arbitration Functions in Cooperative Games with Overlapping Coalitions Subsets of agents can *deviate* from an outcome by withdrawing some or all of their resources from some or all of the partial coalitions they participate in. The *arbitration function* [23] is a mapping $\mathcal{A}$ that receives as its input (a) a feasible outcome $(CS, \mathbf{x})$, (b) a deviating set $S \subseteq N$ and (c) S's proposed deviation, i.e., a list of resources that members of S intend to withdraw from each coalition in CS. Given this data, $\mathcal{A}$ returns a number for each coalition $\mathbf{c}$ with $\mathit{supp}(\mathbf{c}) \cap S \neq \emptyset$, $\mathit{supp}(\mathbf{c}) \cap (N \setminus S) \neq \emptyset$; this number represents how much of $\mathbf{c}$'s payoffs the agents in $S \cap \mathit{supp}(\mathbf{c})$ will be allowed to keep if they deviate. In general, this number may depend on S's behavior outside of $\mathbf{c}$: for instance, $\mathbf{c}$ may be unwilling to pay S if S hurts some players in $\mathit{supp}(\mathbf{c}) \cap (N \setminus S)$ in some other coalition, or it may withhold the payment completely if S deviates in any way whatsoever. We assume that $\mathcal{A}$ is *normalized*: if a set withdraws all of its resources from some partial coalition, it receives nothing from it. We also assume that $\mathcal{A}$ is *deviation-monotone*: the deviators cannot increase the payoff they receive from a partial coalition by withdrawing more resources from it.

We denote by $\mathcal{A}^*(CS, \mathbf{x}, S)$ the most that a set $S \subseteq N$ can get when deviating from $(CS, \mathbf{x})$ under the arbitration function $\mathcal{A}$ (including the payoff from the partial coalitions that the deviators form among themselves). An outcome $(CS, \mathbf{x})$ is said to be in the $\mathcal{A}$*-core*, or $\mathcal{A}$*-stable*, if no set S can deviate (and then share payoffs from the deviation) so that each $i \in S$ gets more than $p_i(CS, \mathbf{x})$, when payoffs to deviators from coalitions with non-deviators are given by $\mathcal{A}$. Zick and Elkind [23] show that an outcome is $\mathcal{A}$-stable if and only if $p_S(CS, \mathbf{x}) \geq \mathcal{A}^*(CS, \mathbf{x}, S)$ for all $S \subseteq N$.

Three types of reactions to set deviations are described in [3]; in the terminology of [23], these are arbitration functions. First, under the *conservative* arbitration function, any coalition $\mathbf{c}$ with $\mathit{supp}(\mathbf{c}) \cap (N \setminus S) \neq \emptyset$ pays nothing to S; this notion of deviation is the most restrictive, and allows a deviating set only the payoff from whatever coalitions it forms on its own. Second, under the *refined* arbitration function, $\mathbf{c}$ allows S to keep its payoff as long as no member of S changes his contribution to $\mathbf{c}$. Third, under the *optimistic* arbitration function, the deviators may keep some of $\mathbf{c}$'s payoff even if they withdraw some resources from $\mathbf{c}$; specifically, if they can ensure that each agent in $\mathit{supp}(\mathbf{c}) \cap (N \setminus S)$ receives as much from the reduced coalition as it did before the deviation, they can keep the remaining payoff. Under the assumptions on arbitration functions given in [23], the payoff given by the optimistic arbitration function is the most that any arbitration function may give. This means that if an outcome is stable w.r.t. the optimistic arbitration function, it is stable under any arbitration function.

3. OUR MODEL

In the model of [3, 23], each agent may divide his resources in any way he chooses. This leads to a variety of conceptual and algorithmic complications: for instance, there is no apriori bound on the size of the coalition structure, and it is not clear how to represent the characteristic function and the arbitration function.

To circumvent these difficulties, we assume that agents may only divide their resources in a discrete manner: each agent $i \in N$ has a positive integer weight w_i, and may allocate an integer part of it to a partial coalition. This is a reasonable assumption in most multiagent settings, where agents allocate hours, money or memory space to tasks. We will refer to such games as *discrete OCF games*. We set $\Psi = \max_{i \in N}\{w_i\}$; note that the case $\Psi = 1$ corresponds to the standard model of characteristic function games, i.e., one that does not admit overlapping coalitions. For our asymptotic bounds, we will assume $\Psi > 1$. Let $\mathbf{w} = (w_1, \dots, w_n)$ and $\mathcal{W} = [0, w_1] \times \dots \times [0, w_n]$.

We can interpret the characteristic function v of a discrete OCF game as a mapping from $\mathcal{W}$ to $\mathbb{R}$ and modify the definition of the superadditive cover and other notions introduced in Section 2 in a similar manner. Given a subset $S \subseteq N$ and some $\mathbf{q} \in \mathcal{W}$, we let $\mathbf{q}^S$ be the vector $\mathbf{q}$ with all coordinates $i \notin S$ set to 0. Also, we define $\mathcal{W}(S) = \{\mathbf{q} \in \mathcal{W} \mid \mathbf{q} \le \mathbf{w}^S\}$. We will assume that the value of each partial coalition is a non-negative rational number that can be encoded using $\mathrm{poly}(n, \log \Psi)$ bits. Under this assumption for any fixed value of Ψ there are finitely many discrete n-player OCF games where the weight of each player is bounded by Ψ, and any such game can be represented by a vector of length $(\Psi + 1)^n$.

The size of this vector representation is exponential in n, which is unacceptable for most applications. We will therefore limit our attention to games where there is an apriori bound on the admissible coalition size. Namely, we say that a game with overlapping coalitions is a *k-OCF game* if $v(\mathbf{q}) = 0$ for any $\mathbf{q} \in \mathcal{W}$ with $|supp(\mathbf{q})| > k$. Clearly, a discrete k-OCF game can be represented using $\binom{n}{k}(\Psi + 1)^k$ values; this number is polynomial in Ψ and n if k is bounded by a constant. In the rest of the paper, we will assume that a k-OCF game is represented by a list that consists of all partial coalitions $\mathbf{q}$ with $|supp(\mathbf{q})| \le k$, together with their values. We will write $v_{a_1,\dots,a_k}(q_1, \dots, q_k)$ to denote the value of the partial coalition with support $\{a_1, \dots, a_k\}$ that receives q_i units of weight from agent a_i, $i = 1, \dots, k$.

An important advantage of this model is that it makes it relatively easy to deal with arbitration functions. Indeed, in a discrete OCF game each coalition structure consists of at most $n(\Psi + 1)$ partial coalitions. This means, in particular, that the input to the arbitration function can be represented using $\mathrm{poly}(n, \Psi, ||\mathbf{x}||)$ bits, where $||\mathbf{x}||$ is the bitsize of the payoff vector $\mathbf{x}$. We will assume that our algorithms have oracle access to the arbitration function; the observation above means that in our model querying this oracle takes time polynomial in the game representation size.

REMARK 3.1. If the characteristic function of a discrete OCF game is efficiently computable, it can be encoded more succinctly by a circuit that takes the vector of agents' resources as its input and outputs the value of the corresponding partial coalition. More formally, this circuit would have $n\lceil\log(\Psi + 1)\rceil$ inputs: the i-th $\lceil\log(\Psi + 1)\rceil$-bit block of inputs would encode the contribution of agent i. Any discrete OCF game with a polynomial-time computable characteristic function $v : \mathcal{W} \to \mathbb{Q}_+$ can be represented by a circuit of size $\mathrm{poly}(n, \log \Psi)$. This representation is succinct even if we do not limit the coalition sizes. However, it has two important disadvantages. First computing v^*, which is the most basic computational problem associated with an OCF game, becomes NP-hard even for $n = 1$; this follows by a straightforward reduction from the UNBOUNDED KNAPSACK problem [15] (a similar reduction, albeit in a slightly different context, can be found in [3]). Second, since the agents may form a coalition structure of size $\Omega(n\Psi)$, querying the arbitration function may be exponentially expensive in this model. Therefore, in what follows, we will not consider this representation.

4. *K*-OCF GAMES: FIRST OBSERVATIONS

The representation of a k-OCF game explicitly provides the value of each partial coalition $\mathbf{q}$. However, if we are interested in computing the total profit that can be earned by agents whose resources are given by $\mathbf{w}$, we need to take into account that these agents may split their resources among several partial coalitions. Thus, we need to compute the value of the superadditive cover v^* on $\mathbf{w}$. This computational problem is formalized as follows.

Name: OPTVAL
Input: A discrete k-OCF game over n players with maximum weight Ψ, a coalition $\mathbf{q} \in \mathcal{W}$, and a value r.
Question: Is $v^*(\mathbf{q}) \ge r$?

It is not hard to show that this problem is tractable if we additionally require that $|supp(\mathbf{q})|$ is bounded by a constant.

PROPOSITION 4.1. *Given a discrete OCF game and a partial coalition $\mathbf{q}$ with $|supp(\mathbf{q})| \le t$, one can compute $v^*(\mathbf{q})$ in time* $\mathrm{poly}(\Psi^t)$.

PROOF. We have $v^*(\mathbf{q}) = \max\{v^*(\mathbf{q} - \mathbf{r}) + v(\mathbf{r}) \mid \mathbf{r} \le \mathbf{q}\}$. Thus, if we have computed $v^*(\mathbf{q}')$ for all $\mathbf{q}' < \mathbf{q}$, we can compute $v^*(\mathbf{q})$ in $\mathcal{O}((\Psi + 1)^t)$ time. Hence, we can compute $v^*(\mathbf{q})$ in time $\mathcal{O}(t(\Psi + 1)^{t+1})$ by dynamic programming. □

However, in general OPTVAL is computationally difficult, even if the maximum coalition size and the maximum weight are bounded by small constants.

THEOREM 4.2. OPTVAL *is* NP*-complete even if* $k \le 2$, $\Psi \le 3$.

PROOF. To see that this problem is in NP, observe that is suffices to guess a coalition structure $CS = (\mathbf{q}^1, \dots, \mathbf{q}^m)$ with $\sum_{j=1}^m q_i^j \le w_i$ and $v(CS) \ge r$; note that the size of this coalition structure is at most $n(\Psi + 1)$, which is polynomial in the input size.

For the hardness proof, we provide a reduction from EXACT COVER BY 3-SETS (X3C) [10]. Recall that an instance of X3C is given by a finite set A, $|A| = 3\ell$, and a collection of subsets $\mathcal{S} = \{S_1, \dots, S_t\} \subseteq 2^A$ such that $|S_j| = 3$ for all $j = 1, \dots, t$. It is a "yes"-instance if A can be covered by exactly ℓ sets from $\mathcal{S}$. Given an instance $(A, \mathcal{S})$ of X3C, we construct a discrete OCF game with $k = 2$, $\Psi = 3$ as follows. We have an agent a_i of weight 1 for every element $i \in A$ and an agent a_S with weight 3 for every $S \in \mathcal{S}$. The characteristic function is defined as follows: $v_{a_i,a_S}(1, 1) = 2$ if $i \in S$, $v_{a_S}(3) = 5$, and the value of every other partial coalition is 0.

Let $S = \{x, y, z\}$ and consider $G_S = \{a_S, a_x, a_y, a_z\}$. Collectively, the agents in G_S can earn 6 if a_S forms a partial coalition with each of a_x, a_y, and a_z, and contributes one unit of weight to each of these coalitions; in any other coalition structure G_S earns at most 5. Hence, $(A, \mathcal{S})$ admits an exact cover if and only if $v^*(\mathbf{q}) \ge 6\ell + 5(t - \ell) = 5t + \ell$. □

Thus, discrete OCF games present a challenge from the computational perspective even if we severely restrict the maximum weight and coalition size. This means that in order to find tractable classes of such games, we must place further constraints on the coalitions that the agents are allowed to form. To identify the appropriate constraints, let us first consider 2-OCF games. Any such game can be naturally identified with a graph: the vertex set of this graph is N; there is an edge between i and j if there exists a partial coalition $\mathbf{q}$ such that $supp(\mathbf{q}) = \{i, j\}$ and $v(\mathbf{q}) > 0$. We will refer to this graph as the *interaction graph* of the game (N, v). Given this perspective, one may wonder if placing constraints on the interaction graph leads to tractable OCF games. In Section 5, we show that this is indeed the case: many computational problems for discrete 2-OCF games become tractable if the interaction graph is a tree. In Section 6, we extend these results to k-OCF games: such games can be associated with hypergraphs, and we show that many—though not all—of the results of Section 5 hold for k-OCF games whose interaction hypergraph has bounded treewidth.

All hardness results for OCF games derived so far stem from the complexity of computing the superadditive cover; therefore, to circumvent them, we place constraints on the characteristic function v. We now show that if we are interested in stability-related questions, we have to place constraints on the arbitration function $\mathcal{A}$ as well. Specifically, recall that $\mathcal{A}^*(CS, \mathbf{x}, S)$ computes the most that a set $S \subseteq N$ can earn when deviating from $(CS, \mathbf{x})$ under the arbitration function $\mathcal{A}$. In other words, when a set of agents S decides whether to deviate from $(CS, \mathbf{x})$, it needs to compute $\mathcal{A}^*(CS, \mathbf{x}, S)$. In the non-overlapping case, a given coalition S can easily decide whether it should deviate: it suffices to compute $v(S)$ and compare it with the payoff that S receives under $(CS, \mathbf{x})$. In contrast, $\mathcal{A}^*$ can be hard to compute even if $n = 2$ and both v and $\mathcal{A}$ are poly-time computable.

THEOREM 4.3. *If there exists a poly-time algorithm that for any discrete OCF game (N, v) any $CS \in \mathcal{CS}(N)$, any $\mathbf{x} \in \mathcal{I}(CS)$ and any $S \subseteq N$ can compute $\mathcal{A}^*(CS, \mathbf{x}, S)$ given oracle access to $\mathcal{A}$, then* P=NP. *This remains true even if the algorithm is only required to work when $|N| = 2$ and both v and $\mathcal{A}$ are poly-time computable.*

PROOF. We will show that if such an algorithm exists, it can be used to solve instances of SET COVER [10]. Recall that an instance of SET COVER is given by a set of elements A, a collection of subsets $\mathcal{S} = \{S_1, \dots, S_t\} \subseteq 2^A$ and $\ell \in \mathbb{N}$; it is a "yes"-instance if A can be covered by at most ℓ sets from $\mathcal{S}$.

Given an instance of SET COVER, consider a 2-player discrete OCF game where $w_1 = w_2 = t + 2$. The valuation function v is defined as follows. Each player gets payoff 1 for each unit of effort he invests in working on his own, i.e., $v(0, x) = v(x, 0) = x$ for $x = 0, \dots, t + 2$. Further, we have $v(1, 1) = 2$, $v(2, 2) = 10(t + 2)$; the value of v on other partial coalitions can be defined arbitrarily. Let $CS = (\mathbf{q}^1, \dots, \mathbf{q}^t, \mathbf{q}^{t+1})$, where $\mathbf{q}^i = (1, 1)$ for $i = 1, \dots, t$ and $\mathbf{q}^{t+1} = (2, 2)$. Let $\mathbf{x} = (\mathbf{x}^1, \dots, \mathbf{x}^{t+1})$ be a payoff vector that allocates all payoff from $\mathbf{q}^1, \dots, \mathbf{q}^t$ to 2 and splits the payoff from $\mathbf{q}^{t+1}$ equally between 1 and 2.

Now, recall that the input to the arbitration function $\mathcal{A}$ is an outcome $(CS, \mathbf{x})$ and a resource withdrawal pattern for the deviating coalition S. Let $(CS, \mathbf{x})$ be as constructed above, $S = \{1\}$, and suppose that player 1 withdraws from partial coalitions $\mathbf{q}^{i_1}, \dots, \mathbf{q}^{i_s}$, where $\{i_1, \dots, i_s\} \subseteq \{1, \dots, t\}$. We define $\mathcal{A}$ so that on this input player 1 receives nothing from $\mathbf{q}^1, \dots, \mathbf{q}^t$; moreover, player 1 keeps his payoff from $\mathbf{q}^{t+1}$ if and only if the collection $\{S_i \mid i \neq i_1, \dots, i_s\}$ is a cover for A; otherwise, player 1 gets nothing from $\mathbf{q}^{t+1}$. We can define $\mathcal{A}$ arbitrarily on other inputs; for concreteness, let us say that it coincides with the refined arbitrator. Note that $\mathcal{A}$ is normalized, deviation-monotone and polynomial-time computable.

If player 1 withdraws x units of resources from $\mathbf{q}^1, \dots, \mathbf{q}^t$, he can use them to earn x by working on his own. Thus, under $\mathcal{A}$ player 1 maximizes his payoff by withdrawing resources from as many coalitions among $\mathbf{q}^1, \dots, \mathbf{q}^t$ as possible, subject to the constraint that the coalitions he still contributes to correspond to a cover of A. Thus, $\mathcal{A}^*(CS, \mathbf{x}, \{1\}) \geq 5(t + 2) + (t - \ell)$ if and only if the input instance of SET COVER admits a cover of size at most ℓ. $\square$

Intuitively, the hardness of computing $\mathcal{A}^*$ stems from the fact that, when determining whether player 1 gets to keep his payoff from $\mathbf{q}^{t+1}$, the arbitration function bases its decision on player 1's global behavior. This motivates the following definition.

DEFINITION 4.4. *An arbitration function $\mathcal{A}$ for an OCF game (N, v) is said to be* local *if the payoff to a deviating set S from a coalition $\mathbf{q}$ is determined by the resources that S takes from $\mathbf{q}$.*

It is easy to see that the conservative, refined and optimistic arbitration functions that were defined in [3] are local. In contrast, the arbitration function used the proof of Theorem 4.3 is non-local. Another example of a non-local arbitration function is the *sensitive arbitrator* defined in [23]: under this arbitration function, the deviating set S keeps its payoff from a partial coalition $\mathbf{q}$ if none of the players in *supp*$(\mathbf{q}) \cap (N \setminus S)$ are hurt by the deviation.

Local arbitration functions are easier to work with, as they do not need to receive the entire coalition structure as their input; thus, a local arbitrator can be queried in polynomial time even assuming the circuit representation discussed in Remark 3.1. Even more importantly, they can be used to circumvent the hardness result of Theorem 4.3.

THEOREM 4.5. *For any discrete OCF game (N, v), given a local arbitration function $\mathcal{A}$, an outcome $(CS, \mathbf{x})$ and a set $S \subseteq N$, one can compute $\mathcal{A}^*(CS, \mathbf{x}, S)$ in time* $\mathrm{poly}(\Psi^{|S|})$.

PROOF. We first observe that a coalition structure CS has at most $(\Psi + 1)|S|$ coalitions that involve players in S. Given a coalition structure CS, let $\mathbf{q}^1, \dots, \mathbf{q}^m$ be the list of partial coalitions that receive contributions from both S and $N \setminus S$. Suppose that for $\ell = 1, \dots, m$, S's contribution to $\mathbf{q}^\ell$ is given by resource vector $\mathbf{r}^\ell \in \mathcal{W}(S)$, such that $\mathbf{r}^\ell \leq \mathbf{q}^\ell$; w.l.o.g., we assume that for each ℓ the vector $\mathbf{r}^\ell$ has at least one strictly positive coordinate. Now, suppose that players in S invest $\mathbf{s} \in \mathcal{W}(S)$ units of resources in partial coalitions among themselves and want to withdraw an additional $\mathbf{t} \in \mathcal{W}(S)$ from CS. They would get $v^*(\mathbf{s} + \mathbf{t})$ from working on their own, plus the most that S can get from the arbitration function, which depends on the coalitions affected by this deviation. Let us denote by $A(\mathbf{y}; \ell)$ the most that the arbitration function will give S if they withdraw $\mathbf{y}$ resources from the first ℓ coalitions, where $1 \leq \ell \leq m$. We also denote by $\mathcal{A}(\mathbf{z}; \ell)$ the payoff to S from coalition $\mathbf{q}^\ell$ if it withdraws $\mathbf{z} \leq \mathbf{r}^\ell$ from $\mathbf{q}^\ell$. We obtain

$$A(\mathbf{z}; \ell) = \max\{A(\mathbf{y}; \ell - 1) + \mathcal{A}(\mathbf{z} - \mathbf{y}; \ell) \mid 0 \leq \mathbf{y} \leq \mathbf{z}\}.$$

This shows that we can compute $A(\mathbf{z}; m)$ in $\mathcal{O}(m(\Psi + 1)^{|S|}) = \mathrm{poly}(\Psi|S|)$ steps. Now, the $\mathcal{A}^*(CS, \mathbf{x}, S)$ can be computed as $\max\{v^*(\mathbf{s} + \mathbf{t}) + A(\mathbf{t}; m) \mid 0 \leq \mathbf{t} \leq \mathbf{w}^S - \mathbf{s}\}$. $\square$

Theorem 4.5 implies that $\mathcal{A}^*$ can be computed in time polynomial in Ψ if $|S|$ is bounded by a constant. From this point onwards, we assume that $\mathcal{A}$ is local, unless explicitly stated otherwise.

5. 2-OCF GAMES ON TREES

In this section, we focus on discrete 2-OCF games where the interaction graph is a tree (all of our results generalize immediately to the case where this graph is a forest). We can root this tree at an arbitrary node; if $r \in N$ is chosen as the root, we denote by $C_i(r)$ the children of player i and by $T_i(r)$ the nodes of the subtree rooted at i. We omit r from the notation when it is clear from the context.

We can efficiently compute v^* for such games.

THEOREM 5.1. *If the interaction graph is a tree, $v^*(\mathbf{w})$ can be computed in time* $\mathrm{poly}(n, \Psi)$.

PROOF. We will show how to compute $v^*(\mathbf{w})$. To extend the algorithm to arbitrary $\mathbf{q}$, we can consider the game with the set of players $supp(\mathbf{q})$ and weights given by $\mathbf{q}$.

We arbitrarily choose some player $r \in N$ to be the root, and process the players starting from the leaves and moving towards the root. For each node i and each $w = 0, \dots, w_i$, let $u_i(w)$ denote the most that the players in T_i could earn if i had weight w. When processing i, we compute the quantities $u_i(0), \dots, u_i(w_i)$ based on the results of similar computations at each of i's children. $v^*(\mathbf{w}) = u_r(w_r)$, so once we reach the root, we output $u_r(w_r)$ and stop.

Consider first a leaf i. We have $u_i(w) = v_i^*(w)$, so, by Proposition 4.1, we can compute all $u_i(w)$, $0 \le w \le w_i$, in time $\mathcal{O}(\Psi^2)$.

Now consider an internal node i. Suppose that $C_i = \{i_1, \dots, i_\ell\}$ and for each $i_j \in C_i$ the quantities $u_{i_j}(w)$ for $w = 0, \dots, w_{i_j}$ have been computed. For $j = 0, \dots, \ell$, let $T_{i,j}$ be the tree obtained from T_i by removing subtrees rooted at $i_{j+1}, \dots, i_\ell$, and let $u_i(z; j)$ be the most that the players in $T_{i,j}$ could earn if i had weight z, $0 \le z \le w_i$. We have $u_i(z; 0) = v_i^*(z)$. Further, having computed $u_i(z'; j-1)$ for all $z' = 0, \dots, w_i$, we can compute $u_i(z; j)$ for all $z = 0, \dots, w_i$ as

$$u_i(z; j) = \max_{\substack{0 \le x \le z \\ 0 \le y \le w_{i_j}}} \{v^*_{i,i_j}(x, y) + u_i(z-x; j-1) + u_{i_j}(w_{i_j} - y)\}.$$

Indeed, players i and i_j need to decide how much weight to allocate to working together. Given this decision, they should optimally allocate their remaining weight to collaboration with, respectively, $i_1, \dots, i_{j-1}$ and T_{i_j}. The expression above optimizes over all choices available to i and i_j.

By Proposition 4.1, $v^*_{i,i_j}(x, y)$ can be computed in time $\mathcal{O}(\Psi^3)$, and $u_i(z-x; j-1)$, $u_{i_j}(w_{i_j} - y)$ have been pre-computed. Thus, for any fixed z this computation takes $\mathcal{O}(\Psi^5)$ steps, and computing all $u_i(z; j)$, $z = 0, \dots, w_j$, for a fixed value of k takes $\mathcal{O}(\Psi^6)$ steps. We clearly have $u_i(w) = u_i(w; \ell)$; hence, i can compute $u_i(0), \dots, u_i(w_i)$ in $\mathcal{O}(|C_i|\Psi^6)$ steps. As this computation has to performed at every internal node, the overall running time of our algorithm is $\sum_{i=1}^n \mathcal{O}(|C_i|\Psi^6) = \mathcal{O}(n\Psi^6)$. □

We now move on to the study of stability-related questions. The first problem we consider is computing $\mathcal{A}^*$, i.e., deciding whether a given coalition can profitably deviate under $\mathcal{A}$. In general, this problem is NP-hard even for discrete 2-OCF games and local arbitrators: this follows from Theorem 4.2, combined with the observation that $\mathcal{A}^*(CS, \mathbf{x}, N) = v^*(\mathbf{w})$ for any outcome $(CS, \mathbf{x})$ and any arbitration function $\mathcal{A}$. However, computing $\mathcal{A}^*$ becomes easy when the interaction graph is a tree.

THEOREM 5.2. *Given a discrete 2-OCF n-player game and a local arbitration function $\mathcal{A}$, if the interaction graph is a tree, we can compute $\mathcal{A}^*(CS, \mathbf{x}, S)$ for any $S \subseteq N$ and any outcome $(CS, \mathbf{x})$ in time* $\text{poly}(n, \Psi)$.

PROOF SKETCH. We use the algorithm given in the proof of Theorem 5.1, with the modification that each of the deviators also has to decide how much weight to keep in his collaboration with non-deviators; this decision is not too difficult since the interactions are between pairs of agents, and the arbitration function is local. In more detail, given an outcome $(CS, \mathbf{x})$ and a deviating set S, we construct a new discrete 2-OCF game where the set of players is S, and the characteristic function $\bar{v}$ is defined so that $\bar{v}_{i,j} \equiv v_{i,j}$ for $i, j \in S$ and $\bar{v}_i^*(w)$ outputs the most that player i can make by allocating w units of weight to working on his own *and with his neighbors from* $N \setminus S$ (this quantity depends on $\mathcal{A}$ and $(CS, \mathbf{x})$). It can be shown that $\bar{v}^*(S)$ can be computed by dynamic programming in time $\text{poly}(n, \Psi)$ and $\mathcal{A}^*(CS, \mathbf{x}, S) = \bar{v}^*(S)$; we omit the full proof due to space constraints. □

We are now ready to present an algorithm for checking whether a given outcome is in the $\mathcal{A}$-core. This problem is closely related to that of computing $\mathcal{A}^*$: an outcome $(CS, \mathbf{x})$ is in the $\mathcal{A}$-core if and only if the *excess* $e(CS, \mathbf{x}, S) = \mathcal{A}^*(CS, \mathbf{x}, S) - p_S(CS, \mathbf{x})$ is non-positive for all coalitions $S \subseteq N$. Thus, we need to check whether there exists a subset $S \subseteq N$ with $e(CS, \mathbf{x}, S) > 0$. Note that it suffices to limit our attention to connected subsets of N: if $e(CS, \mathbf{x}, S) > 0$ and S is not connected, then some connected component S' of S also satisfies $e(CS, \mathbf{x}, S') > 0$.

THEOREM 5.3. *If the interaction graph is a tree, we can verify whether a given outcome $(CS, \mathbf{x})$ is $\mathcal{A}$-stable in time* $\text{poly}(n, \Psi)$.

PROOF. Fix an outcome $(CS, \mathbf{x})$ and set $p_i = p_i(CS, \mathbf{x})$ for all $i \in N$.

Again, we pick an arbitrary $r \in N$ as a root. We say that $S \subseteq N$ is *rooted* at $i \in N$ if $i \in S$ and the members of S form a subtree of T_i. We observe that every set $S \subseteq N$ is rooted at a unique $i \in N$. Given a vertex i, let E_i denote the maximum excess of a set rooted at i. Clearly, $(CS, \mathbf{x})$ is not $\mathcal{A}$-stable if and only if $E_i > 0$ for some $i \in N$. We will now show how to compute E_i for all $i \in N$. We proceed from the leaves to the root, and terminate (and report that $(CS, \mathbf{x})$ is not $\mathcal{A}$-stable) if we discover a vertex i with $E_i > 0$. If $E_i \le 0$ for all $i \in N$, we report that $(CS, \mathbf{x})$ is $\mathcal{A}$-stable.

Given two agents $i, j \in N$, let $w_{i,j}$ denote the weight that i assigns to interacting with j.We will now define two auxiliary values. First, given a neighbor j of i, we define $\alpha_{i,j}(w)$ to be the most that $\mathcal{A}$ will give i if he keeps a total weight of $w \le w_{i,j}$ in the coalitions that he formed with j in $(CS, \mathbf{x})$; by Theorem 4.5, $\alpha_{i,j}(w)$ is computable in time $\text{poly}(\Psi)$. Second, we define $D_i(w)$ to be the maximum excess of a subset rooted at i if i were to contribute w to T_i and nothing to his parent $p(i)$. In this notation,

$$E_i = \max\{D_i(w) + \alpha_{i,p(i)}(y) \mid w + y = w_i, w \ge w_i - w_{i,p(i)}\};$$

the condition $w \ge w_i - w_{i,p(i)}$ ensures that $p(i)$ is not among the deviators. It remains to show how to compute $D_i(w)$ in time $\text{poly}(n, \Psi)$ for all $i \in N$ and $w_i - w_{i,p(i)} \le w \le w_i$.

Consider an agent i with children $C_i = \{i_1, \dots, i_\ell\}$, and suppose that we have computed $D_{i_j}(z)$ for each $i_j \in C_i$ and each z, $w_{i_j} - w_{i_j,i} \le z \le w_{i_j}$ (this encompasses the possibility that i is a leaf, as $C_i = \emptyset$ in that case). For $j = 0, \dots, \ell$, let $T_{i,j}$ be the tree obtained from T_i by removing subtrees rooted at $i_{j+1}, \dots, i_\ell$. Let $D_i(w; j)$ be the maximum excess of a set rooted at i that is fully contained in $T_{i,j}$, assuming that i contributes w to $T_{i,j}$ and nothing to his parent or his children $i_{j+1}, \dots, i_\ell$; we have $D_i(w) = D_i(w; \ell)$. We will compute $D_i(w; j)$ by induction on j.

We have $D_i(w; 0) = v_i^*(w) - p_i$ for all $w = w_i - w_{i,p(i)}, \dots, w_i$. Now, consider $j > 0$. Agent i can either include i_j in the deviating set or deviate (partially or fully) from the coalitions that it forms with i_j in $(CS, \mathbf{x})$. Thus, $D_i(w; j) = \max\{D_1, D_2\}$, where

$$D_1 = \max_{\substack{y=0,\dots,w \\ z=0,\dots,w_{i_j}}} \{D_i(y; j-1) + v^*_{i,i_j}(w-y, z) + D_j(w_{i_j} - z)\}.$$

and

$$D_2 = \max_{z=0,\dots,w_{i,i_j}} \{D_i(w-z; j-1) + \alpha_{i,i_j}(z)\}.$$

Thus, we can efficiently compute $D_i(w; j)$, and hence also $D_i(w)$ and E_i. □

We have shown how to check whether a specific outcome is in the $\mathcal{A}$-core. We can use this algorithm as a subroutine to check whether the $\mathcal{A}$-core is non-empty. Specifically, we can go over all coalition structures in $\mathcal{CS}$, and, for each $CS \in \mathcal{CS}$, check if there exists a payoff vector $\mathbf{x} \in \mathcal{I}(CS)$ such that $(CS, \mathbf{x})$ is $\mathcal{A}$-stable: the conditions on $\mathbf{x}$ can be encoded by a linear program that, despite being exponential in size, admits a polynomial-time separation oracle, namely, the one constructed in Theorem 5.3. This implies that this linear program can be solved in polynomial time [19]; we omit the details of this argument due to space constraints. However, enumerating all candidate coalition structures is prohibitively expensive. We will now argue that, at least for the conservative core, this is not necessary: we will show how to explicitly construct an outcome in the conservative core of a discrete 2-OCF game on a tree.

Consider a coalition structure CS such that $v(CS) = v^*(\mathbf{w})$. For any $i, j \in N$ that are connected by an edge, we denote by $w_{i,j}$ the amount of weight that i devotes to interacting with j under CS; note that $w_{j,i}$ need not be equal to $w_{i,j}$. If we remove the edge (i, j), our tree splits into two trees: we will denote the vertex sets of these trees by $V_{i,j}$ and $V_{j,i}$, respectively (where $i \in V_{i,j}$, $j \in V_{j,i}$). We have

$$\begin{aligned} v^*(\mathbf{w}) &= v^*_{i,j}(w_{i,j}, w_{j,i}) + v^*(\mathbf{w}^{V_{i,j}} - w_{i,j}\mathbf{e}^{\{i\}}) \\ &+ v^*(\mathbf{w}^{V_{j,i}} - w_{j,i}\mathbf{e}^{\{j\}}). \end{aligned} \quad (1)$$

We observe that i and j can divide the value $v^*_{i,j}(w_{i,j}, w_{j,i})$ between themselves in any way they wish. Indeed, there is a coalition structure $CS_{i,j} \in \mathcal{CS}(\{i, j\})$ such that $v^*_{i,j}(w_{i,j}, w_{j,i}) = v(CS_{i,j})$, and every coalition in $CS_{i,j}$ receives contributions from both i and j.

We will now derive some constraints on the outcomes in the conservative core.

PROPOSITION 5.4. *If $(CS, \mathbf{x})$ is in the conservative core, then the total payoff to i from interacting with j is at least $v^*(\mathbf{w}^{V_{i,j}}) - v^*(\mathbf{w}^{V_{i,j}} - w_{i,j}\mathbf{e}^{\{i\}})$ and at most $v^*_{i,j}(w_{i,j}, w_{j,i}) - v^*(\mathbf{w}^{V_{i,j}}) + v^*(\mathbf{w}^{V_{i,j}} - w_{i,j}\mathbf{e}^{\{i\}})$.*

PROOF. Since $v^*(\mathbf{w}^{V_{i,j}}) + v^*(\mathbf{w}^{V_{j,i}}) \leq v^*(\mathbf{w})$, from (1) we obtain $v^*(\mathbf{w}^{V_{i,j}}) - v^*(\mathbf{w}^{V_{i,j}} - w_{i,j}\mathbf{e}^{\{i\}}) \leq v^*_{i,j}(w_{i,j}, w_{j,i}) - v^*(\mathbf{w}^{V_{j,i}}) + v^*(\mathbf{w}^{V_{j,i}} - w_{j,i}\mathbf{e}^{\{j\}})$.

If i gets less than $v^*(\mathbf{w}^{V_{i,j}}) - v^*(\mathbf{w}^{V_{i,j}} - w_{i,j}\mathbf{e}^{\{i\}})$ from interacting with j, then the total payoff to $V_{i,j}$ is less than $v^*(\mathbf{w}^{V_{i,j}})$, a contradiction with $(CS, \mathbf{x})$ being in the conservative core.

Similarly, if i gets more than $v^*_{i,j}(w_{i,j}, w_{j,i}) - v^*(\mathbf{w}^{V_{j,i}}) + v^*(\mathbf{w}^{V_{j,i}} - w_{j,i}\mathbf{e}^{\{j\}})$ from interacting with j, then j gets less than $v^*(\mathbf{w}^{V_{j,i}}) - v^*(\mathbf{w}^{V_{j,i}} - w_{j,i}\mathbf{e}^{\{j\}})$ from their interaction, which implies that the total payoff to $V_{j,i}$ is less than $v^*(\mathbf{w}^{V_{j,i}})$, a contradiction. □

Now, consider a coalition structure CS with $v^*(\mathbf{w}) = v(CS)$. We now show how to assign payoffs to agents so that the resulting outcome is in the conservative core; in doing so, we are guided by Proposition 5.4. Note that we do not construct a pre-imputation $\mathbf{x} \in \mathcal{I}(CS)$ explicitly. Rather, we simply indicate the cumulative payments to the agents: this is sufficient as long as we are only interested in the conservative core.

THEOREM 5.5. *For any discrete 2-OCF game $G = (N, v)$ whose interaction graph is a tree and any coalition structure $CS \in \mathcal{CS}(N)$ such that $v^*(\mathbf{w}) = v(CS)$, there is a payoff vector $\mathbf{x} \in \mathcal{I}(CS)$ such that $(CS, \mathbf{x})$ is in the conservative core.*

PROOF. Let CS be a coalition structure such that $v^*(\mathbf{w}) = v(CS)$. Let $r \in N$ be the root. Recall that T_i is the set of vertices of the tree rooted in i, i.e., if p is the parent of i then $T_i = V_{i,p}$. We allocate to agent $i \in N$:

- all payoff from coalitions he forms on his own;
- $v^*(\mathbf{w}^{T_i}) - v^*(\mathbf{w}^{T_i} - w_{i,p}\mathbf{e}^{\{i\}})$ from the interaction with his parent p (assuming $i \neq r$);
- $v^*_{i,j}(w_{i,j}, w_{j,i}) - v^*(\mathbf{w}^{T_i}) + v^*(\mathbf{w}^{T_i} - w_{i,j}\mathbf{e}^{\{i\}})$ from the interaction with each of his children $j \in C_i$.

This payoff division is feasible and efficient: the payoff from every edge (i, j) is split between i and j. Thus, there exists a pre-imputation $\mathbf{x} \in \mathcal{I}(CS)$ supporting these payoffs. It remains to show that the resulting outcome $(CS, \mathbf{x})$ is stable with respect to the conservative arbitrator. We will require the following lemma.

LEMMA 5.6. *Under the outcome $(CS, \mathbf{x})$, the total payoff to agents in $V_{i,j}$ is exactly $v^*(\mathbf{w}^{V_{i,j}})$, for any $i \in N$.*

PROOF. The lemma is clearly true if $i = r$. If $i \neq r$, let p be the parent of i. Agent i contributes weight $w_i - w_{i,p}$ to T_i. Since CS is an optimal coalition structure, agents in T_i earn $v^*(\mathbf{w}^{T_i} - w_{i,p}\mathbf{e}^{\{i\}})$, which they share among themselves. Further, i also receives $v^*(\mathbf{w}^{T_i}) - v^*(\mathbf{w}^{T_i} - w_{i,p}\mathbf{e}^{\{p\}})$ from his parent. Together, this adds up to $v^*(\mathbf{w}^{T_i})$. □

Now, suppose that a subset of agents S can profitably deviate from $(CS, \mathbf{x})$ by forming some $CS' \in \mathcal{CS}(S)$; assume that S is rooted at i. Let R consist of all vertices in $T_i \setminus S$ whose parents belong to S. Note that we have $T_i = S \cup (\cup_{j \in R} T_j)$. Now, consider a coalition structure over T_i where for each $j \in R$ the agents in T_j form the optimal coalition structure among themselves, and agents in S form CS'. By Lemma 5.6, in this new coalition structure the value of each T_j, $j \in R$, is the same as its payoff in $(CS, \mathbf{x})$. On the other hand, since S can profitably deviate using CS', $v(CS') > p_S(CS, \mathbf{x})$. We conclude that in this coalition structure the agents in T_i earn more than in $(CS, \mathbf{x})$. However, by Lemma 5.6 their total payoff in $(CS, \mathbf{x})$ is exactly $v^*(\mathbf{w}^{T_i})$, which is a contradiction. □

Theorem 5.5 does not hold for other arbitration functions, as the following example shows.

EXAMPLE 5.7. Consider a 3 player game where $w_1 = 2, w_2 = 2, w_3 = 1$. Also, $v_1(1) = 5, v_{1,2}(1, 1) = 10, v_{2,3}(1, 1) = 9$. The rest of the valuations are set to 0. One can verify that the refined core of this game is empty.

Note also that the payoff division proposed in Theorem 5.5 depends of the choice of the root, with nodes that are closer to the root reaping the benefits from the collaboration with their children. As a result, this payoff division scheme is not particularly "fair", as players who contribute equally to an interaction may not be paid equally. Consider for example a two-agent setting where both agents have a weight 1 and the value of their interaction is 1. While, intuitively, both players have equal claim to the profit, only one of them will get the payoff, while the other receives nothing.

6. INTERACTION HYPERGRAPHS WITH BOUNDED TREEWIDTH

While 2-OCF games correspond to graphs, k-OCF games with $k > 2$ can be modeled as *hypergraphs*, whose hyperedges are of size at most k: the vertex set of this hypergraph in N and there is an edge $E \subseteq N$ if and only if $v(\mathbf{q}) > 0$ for some $\mathbf{q} \in \mathcal{W}$ with $supp(\mathbf{q}) = E$. The resulting hypergraph is called the *interaction hypergraph* of the corresponding k-OCF game.

Several NP-hard combinatorial optimization problems on hypergraphs become tractable for hypergraphs whose treewidth is known to be bounded by a constant [18]. Problems in cooperative game theory are no exception: Ieong and Shoham [13] and Greco et. al [12] show that when a certain graphical representation of a cooperative game has bounded treewidth, several computational problems (e.g. deciding if the core is not empty) become tractable. Thus, it is only natural to ask whether our previous tractability results for 2-OCF games on trees can be extended to cases where the treewidth of the interaction hypergraphs in question is bounded by a constant. The answer appears to be mostly positive.

We begin by formally introducing the notion of *treewidth* of a hypergraph $H = (V, \mathcal{E})$ as given by Gottlob et al. [11]; this definition is based on the original notion of treewidth given in [18]. Given a hypergraph H, a *tree decomposition* of H is a tree $\mathcal{T}$ with node set $V(\mathcal{T})$ and edge set $E(\mathcal{T})$ such that each node $X \in V(\mathcal{T})$ is a non-empty subset $X \subseteq V$ of vertices of H. We require that if $E \in \mathcal{E}$, then there is some $X \in V(\mathcal{T})$ such that $E \subseteq X$. Moreover, the nodes are required to have the *running intersection property*: if $z \in X \cap Y$, then all nodes Z that are on the path between X and Y contain z as well. The *width* of a tree decomposition $\mathcal{T}$, denoted $\omega(\mathcal{T})$, equals $\max_{X \in V(\mathcal{T})}\{|X| - 1\}$. The *treewidth* of H, $tw(H)$, is the minimum of $\omega(\mathcal{T})$ over all tree decompositions of H. If $tw(H) = d$, then a tree decomposition of H with width d can be found in $\mathcal{O}(|H|^d)$ time.

Given a tree decomposition $\mathcal{T}$ of H with node set $V(\mathcal{T})$ and two nodes $X, Y \in V(\mathcal{T})$, we can associate the edge between X and Y with the set $X \cap Y$. Note that $X \cap Y$ is not empty if $(X, Y) \in E(\mathcal{T})$. Given a subtree $\mathcal{T}'$ of $\mathcal{T}$, we define $N(\mathcal{T}')$ to be $\bigcup_{X \in V(\mathcal{T}'} X$. We will now show how to adapt the proofs of Theorems 5.1, 5.2, and 5.3 for hypergraphs with bounded treewidth.

THEOREM 6.1. *Given a k-OCF game (N, v) whose interaction hypergraph admits a tree decomposition $\mathcal{T}$ of width d and a partial coalition $\mathbf{q} \in \mathcal{W}$, we can compute $v^*(\mathbf{q})$ in time* $\mathrm{poly}(n, \Psi^{d+1})$.

PROOF. We will give the proof for the case $\mathbf{q} = \mathbf{w}$; the general case can be handled similarly (see the proof of Theorem 5.1). We pick one of the sets $R \in V(\mathcal{T})$ to be the root of $\mathcal{T}$. Given a node $X \in V(\mathcal{T})$, we denote by $\mathcal{T}_X$ the subtree of $\mathcal{T}$ that is rooted at X and by $p(X)$ the parent of X in $\mathcal{T}$ (if $X = R$, we assume $p(X) = X$). For every vector $\mathbf{q} \in \mathcal{W}(X \cap p(X))$, we denote by $\mathcal{T}_X(\mathbf{r})$ the tree $\mathcal{T}_X$ with $X \cap p(X)$ devoting $\mathbf{r}$ to interacting with $\mathcal{T}_X$; note that $|X \cap p(X)| \leq d$. Let $u_X(\mathbf{r})$ denote the most that $N(\mathcal{T}_X(\mathbf{r}))$ can make; clearly, we have $v^*(\mathbf{w}) = u_R(\mathbf{w})$.

We will now show how to compute $u_X(\mathbf{q})$ for each node X and each $\mathbf{q} \in \mathcal{W}(X \cap p(X))$. As in the proof of Theorem 5.1, we proceed by dynamic programming, starting from the leaves and terminating at R. Fix a node X, and let $\mathcal{C}_X = \{C_1, \dots, C_\ell\}$ be the set of X's children; we denote by $u_X(\mathbf{q}; j)$ the most that $N(\mathcal{T}_X)$ can make if X devotes $\mathbf{q}$ to interacting with $C_1, \dots, C_j$ and none to the rest of its children. We have $u_X(\mathbf{q}; 0) = v^*(\mathbf{q})$; this quantity can be computed in time $\mathcal{O}((\Psi + 1)^d)$ by Theorem 4.1. Further, $u_X(\mathbf{q}; j)$ is given by

$$\max\{u_X(\mathbf{q} - \mathbf{y}; j - 1) + u_{C_j}(\mathbf{y}) \mid \mathbf{y} \in \mathcal{W}(C_j \cap X); \mathbf{y} \leq \mathbf{q}\}.$$

The requirement that $\mathbf{y} \leq \mathbf{q}$ is necessary, as $C_j \cap p(X)$ may be non-empty, in which case the amount that $C_j \cap X$ can give to $\mathcal{T}_{C_j}$ is limited by its previous commitment to the parent of X. Hence, we can compute $u_X(\mathbf{q}) = u_X(\mathbf{q}; \ell)$ in time linear in $|\mathcal{C}_X|$ and polynomial in Ψ^{d+1}; summing over all nodes of $\mathcal{T}$, we obtain the desired bound on the running time. □

We can use similar techniques to compute the most that a set can get by $\mathcal{A}$-deviating from some outcome $(CS, \mathbf{x})$.

THEOREM 6.2. *Given a k-OCF game, an outcome $(CS, \mathbf{x})$ and a set $S \subseteq N$ such that the interaction graph induced by S has treewidth d, we can compute $\mathcal{A}^*(CS, \mathbf{x}, S)$ in* $\mathrm{poly}(n, \Psi^{d+1})$ *time.*

PROOF SKETCH. We denote by $\alpha_L(\mathbf{w})$ the most that $\mathcal{A}$ will give a subset $L \subseteq S$ of size at most $d + 1$ if it decides to leave $\mathbf{q} \in \mathcal{W}(L)$ of its weight allocated to non-S members. Since the support of any coalition contains at most $d + 1$ players, computing $\alpha_L(\mathbf{w})$ can be done in a similar manner to Theorem 5.2. Now, given a tree decomposition of S with width d, we again replace the most that any subset $L \subseteq S$ with $|L| \leq d + 1$ can make with the value $\bar{v}^*(\mathbf{q}) = \max\{v^*(\mathbf{x} + \mathbf{y}) + \alpha_i(\mathbf{q} - \mathbf{y}) \mid 0 \leq \mathbf{y} \leq \mathbf{q}\}$ and repeat the computation described in Theorem 6.1. Correctness holds for similar reasons to those described in Theorem 5.2. □

We can also provide an analogue to Theorem 5.3; we omit the proof due to space constraints.

THEOREM 6.3. *Given a k-OCF game whose interaction hypergraph H has treewidth at most d, we can check if an outcome $(CS, \mathbf{x})$ is in the $\mathcal{A}$-core in time* $\mathrm{poly}(n, \Psi^{d+1})$.

Using Theorem 6.3, we obtain the following corollary.

COROLLARY 6.4. *Suppose that the treewidth of the interaction hypergraph of a discrete k-OCF game is at most d. Then, given a coalition structure CS and an arbitration function $\mathcal{A}$, we can check in time* $\mathrm{poly}(n, \Psi^{d+1})$ *if there exists an imputation $\mathbf{x}$ such that $(CS, \mathbf{x})$ is $\mathcal{A}$-stable (and output $\mathbf{x}$ if it exists).*

Briefly, this problem can be encoded as a linear program. Even though this program has exponentially many constraints, it can be solved in time $\mathrm{poly}(n, \Psi^{d+1})$ using the algorithm described in the proof of Theorem 6.3 as a separation oracle.

Finally, we remark that even the conservative core of OCF games with bounded treewidth may be empty. Thus, an analogue of Theorem 5.5 does not hold.

EXAMPLE 6.5. Consider a 2-OCF game with $N = \{1, 2, 3\}$ and $w_i = 1$ for all $i \in N$. Set $v_{i,j}(1, 1) = 1$ for any $i \neq j \in N$, and suppose that $v \equiv 0$ for all other partial coalitions; this is essentially the classic *3-player majority game* [2], which is known to have an empty core. The argument for the classic case can be adapted to show that the conservative core of our game is empty.

7. RELATED WORK

Our work builds directly on the overlapping coalition formation framework of [3, 23]. The main difference between our model and that of [3, 23] is the assumption that the agents' resources are discrete; however, all theoretical results proven in these papers can be shown to hold for the discretized setting.

Restricting the size of admissible coalitions to ensure tractability is a fairly standard approach, see, e.g., the classic work of Shehory and Kraus [20]. More recently, Shrot et al. [22] and Chitnis et al. [5] investigated the parameterized complexity of (non-overlapping) coalitional games, with the maximum coalition size as a parameter. They show that, in the absence of additional constraints on the characteristic function, restricting the coalition size is insufficient for tractability; this is consistent with our results (Theorem 4.2).

Many of our tractability results rely on restricting the interaction between the agents to trees and tree-like structures. This bears close similarity to models in classic cooperative game theory that limit agent interaction. The first such model, proposed by Myerson [17], describes cooperative games where agent interaction is limited by an underlying graph structure; in this model, a coalition may form only if it corresponds to a connected subgraph. Cooperative games on graphs have been subsequently studied by a number of authors; see, e.g., [7, 9, 16, 17]. In particular, Demange [7] describes cooperative games where agents form a hierarchical tree structure and proposes an algorithm that computes a core allocation in this setting; our analysis of 2-OCF games on trees is somewhat similar to this work. However, deriving results in the OCF model is significantly more complicated than in the non-overlapping setting, and the algorithm of [7] cannot be applied directly to our model. Section 6 is inspired by the work of Ieong and Shoham [13] and Greco et al. [12], who analyze the complexity of core-related solution concepts for interaction graphs with bounded treewidth in the non-overlapping setting.

Recently, Anshelevitz and Hoefer [1] introduced *network contribution games*: in these games, each agent has a weight that he may divide among his neighbors, and the value of an interaction depends on the weight each agent devotes to the edge. Their analysis differs from ours in that they assume that the payoff from an edge is divided equally between the agents and study the resulting *non-cooperative* game.

Finally, we remark that the study of computational aspects of coalitional games is a well-established research topic, which received a significant amount of attention in recent years; see, e.g., [4]. Our work makes the first step towards extending this analysis to OCF games.

8. CONCLUSIONS AND FUTURE WORK

Finding optimal coalition structures and stable outcomes are key issues in the analysis of OCF games; we show that these problems are hard in general, but formulate several conditions that make them tractable. We mostly focus on 2-OCF games and acyclic agent interaction graphs; however, we show that our results extend to k-OCF games with constant k and interaction (hyper-)graphs with bounded treewidth.

While our work focuses on achieving computational efficiency by restricting agent interaction, one can also obtain tractability results for OCF games by other means. A natural way of doing so is to extend existing representation languages for non-overlapping coalitional games, such as, e.g., MC-nets [13]—and the algorithms for them—to the OCF setting; the analysis of threshold task games in [3] can be viewed as an example of this approach.

Another way of dealing with hardness results is by designing *approximation* algorithms, i.e., procedures that output a coalition structure that is almost optimal and/or stable. Designing such algorithms (or proving hardness of approximation results) is a fruitful direction for future research.

We have focused mostly on one solution concept: the arbitrated core. Other solution concepts, such as the arbitrated nucleolus, have been proposed and analyzed in [23]. It would be interesting to analyze the computational complexity of finding a nucleolus outcome, or the Shapley value of an agent in cooperative games with overlapping coalitions.

Acknowledgements

This research was supported by the National Research Foundation (Singapore) under grant 2009-08, by NTU SUG (Edith Elkind), and by SINGA graduate fellowship (Yair Zick).

9. REFERENCES

[1] E. Anshelevich and M. Hoefer. Contribution games in social networks. In *ESA'10*, pages 158–169, 2010.

[2] R. Brânzei, D. Dimitrov, and S. Tijs. *Models in cooperative game theory*. Springer, 2005.

[3] G. Chalkiadakis, E. Elkind, E. Markakis, M. Polukarov, and N. Jennings. Cooperative games with overlapping coalitions. *Journal of AI Research*, 39:179–216, 2010.

[4] G. Chalkiadakis, E. Elkind, and M. Wooldridge. *Computational Aspects of Cooperative Game Theory*. Morgan and Claypool, 2011.

[5] R. H. Chitnis, M. T. Hajiaghayi, and V. Liaghat. Parameterized complexity of problems in coalitional resource games. In *AAAI'11*, pages 620–626, 2011.

[6] V. D. Dang, R. K. Dash, A. Rogers, and N. R. Jennings. Overlapping coalition formation for efficient data fusion in multi-sensor networks. In *AAAI'06*, pages 635–640, 2006.

[7] G. Demange. On group stability in hierarchies and networks. *Journal of Political Economy*, 112(4):754–778, 2004.

[8] X. Deng and C. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 19(2):257–266, 1994.

[9] U. Faigle. Cores of games with restricted cooperation. *Mathematical Methods of Operations Research*, 33(6):405–422, 1989.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractibility*. W. H. Freeman and Company, 1979.

[11] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions: A survey. In *MFCS'01*, pages 37–57, 2001.

[12] G. Greco, E. Malizia, L. Palopoli, and F. Scarcello. On the complexity of core, kernel, and bargaining set. *Artificial Intelligence*, 175(12–13):1877–1910, 2011.

[13] S. Ieong and Y. Shoham. Marginal contribution nets: a compact representation scheme for coalitional games. In *ACM EC'05*, pages 193–202. ACM, 2005.

[14] V. Lesser. Cooperative multiagent systems: A personal view of the state of the art. *IEEE Trans. on Knowl. and Data Eng.*, 11:133–142, 1999.

[15] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, 1990.

[16] R. Meir, J. S. Rosenschein, and E. Malizia. Subsidies, stability, and restricted cooperation in coalitional games. In *IJCAI'11*, pages 301–306, 2011.

[17] R. Myerson. Graphs and cooperation in games. *Mathematics of Operations Research*, 2(3):225–229, 1977.

[18] N. Robertson and P. D. Seymour. Graph minors. III. Planar tree-width. *J Comb Theory, B*, 36(1):49–64, 1984.

[19] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, Chichester, 1986.

[20] O. Shehory and S. Kraus. Task allocation via coalition formation among autonomous agents. In *IJCAI'95*, pages 655–661, 1995.

[21] O. Shehory and S. Kraus. Formation of overlapping coalitions for precedence-ordered task-execution among autonomous agents. In *ICMAS'96*, pages 330–337, 1996.

[22] T. Shrot, Y. Aumann, and S. Kraus. Easy and hard coalition resource game formation problems: a parameterized complexity analysis. In *AAMAS'09*, pages 433–440, 2009.

[23] Y. Zick and E. Elkind. Arbitrators in overlapping coalition formation games. In *AAMAS'11*, pages 55–62, 2011.

Handling Negative Value Rules in MC-net-based Coalition Structure Generation

Suguru Ueda[1], Takato Hasegawa[1], Naoyuki Hashimoto[1], Naoki Ohta[2],
Atsushi Iwasaki[1], and Makoto Yokoo[1]
[1]Kyushu University, Fukuoka, 819-0395, Japan
{ueda@agent., hasegawa@agent., hashimoto@agent., iwasaki@, yokoo@}inf.kyushu-u.ac.jp
[2]Ritsumeikan University, Shiga, 525-8577, Japan n-ohta@fc.ritsumei.ac.jp

ABSTRACT

A Coalition Structure Generation (CSG) problem involves partitioning a set of agents into coalitions so that the social surplus is maximized. Recently, Ohta *et al.* developed an efficient algorithm for solving CSG assuming that a characteristic function is represented by a set of rules, such as marginal contribution networks (MC-nets).

In this paper, we extend the formalization of CSG in Ohta *et al.* so that it can handle negative value rules. Here, we assume that a characteristic function is represented by either MC-nets (without externalities) or embedded MC-nets (with externalities). Allowing negative value rules is important since it can reduce the efforts for describing a characteristic function. In particular, in many realistic situations, it is natural to assume that a coalition has negative externalities to other coalitions.

To handle negative value rules, we examine the following three algorithms: (i) a full transformation algorithm, (ii) a partial transformation algorithm, and (iii) a direct encoding algorithm. We show that the full transformation algorithm is not scalable in MC-nets (the worst-case representation size is $\Omega(n^2)$, where n is the number of agents), and does not seem to be tractable in embedded MC-nets (representation size would be $\Omega(2^n)$). In contrast, by using the partial transformation or direct encoding algorithms, an exponential blow-up never occurs even for embedded MC-nets. For embedded MC-nets, the direct encoding algorithm creates less rules than the partial transformation algorithm.

Experimental evaluations show that the direct encoding algorithm is scalable, i.e., an off-the-shelf optimization package (CPLEX) can solve problem instances with 100 agents and rules within 10 seconds.

Categories and Subject Descriptors

I.2.11 [**ARTIFICIAL INTELLIGENCE**]: Distributed Artificial Intelligence – Multiagent systems

General Terms

Algorithms, Theory

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

Keywords

coalition structure generation, constraint optimization, cooperative games

1. INTRODUCTION

Coalition formation is an important capability in automated negotiation among self-interested agents. Coalition Structure Generation (CSG) involves partitioning a set of agents so that social surplus is maximized. This problem has become a popular research topic in AI and multi-agent systems. Solving CSG is equivalent to the *complete set partitioning problem* [17], and various algorithms for solving CSG have been developed [2, 9, 11, 12, 13, 15, 17]. Also, the computational complexity of CSG in various domains has been analyzed [1, 16].

Another active research area in the agent research community is compact representation schemes of a characteristic function. If we naively represent a characteristic function as a table, we require $\Theta(2^n)$ numbers, where n is the number of agents. When the number of agents becomes large, we need a compact method to represent a characteristic function. The main idea of compact representation schemes is to use a set of rules to represent a characteristic function. These compact representation schemes include marginal contribution networks (MC-nets) [5, 7], Synergy Coalition Groups (SCG) [4], etc. The computational complexity for finding various solution concepts when a characteristic function is represented by these compact representation schemes is analyzed in [6].

Recently, Ohta *et al.* [10] introduces an innovative direction for solving CSG by utilizing these compact representation schemes. More specifically, they show that a CSG problem can be formalized as a problem of finding the subset of rules that maximizes the sum of rule values under certain constraints. They also develop mixed integer programming (MIP) formulations of the above optimization problem and experimentally showed that this approach is far more scalable than traditional approaches, e.g., it can solve instances with 120 agents/rules in less than 20 seconds.

In this paper, we extend the formalization of CSG in [10] to handle negative value rules. We concentrate on MC-net-based representations since this representation scheme is more compact and natural than other representation schemes. Also, it can be easily extended to handle externalities among coalitions, i.e., a coalition can affect the performance of other coalitions. This extended representation scheme is called *embedded MC-nets* [8]. Although a rule may have a negative value in the original definition of [7], rule values are restricted to be positive to make the optimization problem simpler in [10].

Although any characteristic function can be represented without using negative value rules (as long as no coalition has a negative value), this restriction can make the representation size of a problem significantly larger. Also, allowing negative value rules can reduce the efforts for describing a characteristic function. Assume the president of a company is trying to reorganize the grouping of workers to maximize the productivity of the company. To utilize CSG techniques, the president needs to represent her knowledge about the characteristic function. When using MC-nets, we can assume most of default situations can be concisely represented by using positive value rules only. Negative value rules would be useful for describing some exceptional situations. Furthermore, to represent externalities among coalitions, the externalities can be either positive or negative.

However, handling negative value rules is a challenging task. If we simply add negative value rules, the MIP formulation in [10] cannot properly find an optimal coalition structure. When all rules have positive values, choosing a rule never hurts. Thus, in the MIP formulation, we construct a solver so that it tries to choose as many rules as possible. The constraints only specify the conditions where rules cannot be selected at the same time. Thus, if we simply include a negative value rule, the solver just ignores this rule, since choosing it *hurts*. We must describe the condition where the solver is forced to choose this negative value rule as a result of choosing other positive rules. Such a condition involves interaction among multiple rules, which is difficult to handle efficiently.

In this paper, we develop following three alternative algorithms to handle negative value rules: (i) a full transformation algorithm, which transforms all negative value rules into positive value rules, (ii) a partial transformation algorithm, which transforms all negative value rules into positive value rules and some negative rules that have a special form, and (iii) a direct encoding algorithm. which creates a set of *dummy rules* so that negative value rules are handled appropriately.

We show that the full transformation algorithm is not scalable in MC-nets, i.e., there exists an instance where the number of newly generated rules becomes $\Omega(n^2)$. Here, n is the number of agents. Furthermore, we show that when this transformation algorithm is applied to embedded MC-nets, there exists an instance where the number of newly generated rules becomes $\Omega(2^n)$. Although we have not yet proved that this exponential blowup is really inevitable, the current results are very negative. Since the CSG algorithm presented in [10] is exponential in the number of rules, even the increase of $\Theta(n^2)$ can be prohibitive.

In contrast, by using the partial transformation or direct encoding algorithms, an exponential blow-up never occurs even for embedded MC-nets. For embedded MC-nets, the direct encoding algorithm creates less rules than the partial transformation algorithm.

We experimentally compare the full/partial transformation algorithms and the direct encoding algorithm, and show that the direct encoding algorithm is by far superior. Also, this algorithm is scalable, i.e., an off-the-shelf optimization package (CPLEX) can solve problem instances with 100 agents/rules within 10 seconds.

2. MODELS/EXISTING WORKS

2.1 Characteristic Function Game

Let A be the set of agents, where $|A| = n$. We assume a characteristic function game, i.e., the value of coalition S is given by a characteristic function $v : 2^A \to \mathbb{R}$. Without loss of generality, we assume $\forall S \subseteq A$, $v(S) \geq 0$ holds. As shown in [15], even if some coalition's values are negative, as long as each coalition's value is bounded (i.e., not infinitely negative), we can normalize the coalition values so that all values are non-negative. This rescaled game is strategically equivalent to the original game. To save space, where there is no risk of confusion, we omit commas when listing sets, for example, writing $\{ab\}$ as a shorthand for $\{a, b\}$.

Coalition Structure Generation (CSG) involves partitioning a set of agents into coalitions so that social surplus is maximized. A coalition structure $CS = \{S_1, S_2, \ldots\}$ is a partition of A and is divided into disjoint and exhaustive coalitions, i.e., $CS = \{S_1, S_2, \ldots\}$ satisfies the following conditions:

$$\forall i, j (i \neq j), S_i \cap S_j = \emptyset, \bigcup_{S_i \in CS} S_i = A.$$

We denote by $\Pi(A)$ the space of all coalition structures over A. The value of coalition structure CS, denoted as $V(CS)$, is given by $V(CS) = \sum_{S_i \in CS} v(S_i)$.

An optimal coalition structure CS^* is a coalition structure that satisfies the following condition: $\forall CS \in \Pi(A), V(CS^*) \geq V(CS)$.

Definition 1 (MC-nets). *An* MC-net *consists of set of* rules R. *Each rule $r \in R$ is of the form: $(L_r) \to v_r$, where L_r is a condition of this rule, which is the conjunctions of literals over A, i.e., $a_1 \wedge \cdots \wedge a_k \wedge \neg a_{k+1} \wedge \cdots \wedge \neg a_m$. We call $\{a_1, \ldots, a_k\}$ positive literals and $\{a_{k+1}, \ldots, a_m\}$ negative literals. We say rule r is* applicable *to coalition S if L_r is true when the values of all Boolean variables that correspond to the agents in S are set to true, and the values of all Boolean variables that correspond to agents in $A \setminus S$ are set to false, i.e., $\bigwedge_{a \in S} a \wedge \bigwedge_{b \in A \setminus S} \neg b \models L_r$ holds. Without loss of generality, we assume each rule has at least one positive literal*[1].

In MC-nets, the condition of a rule must be the conjunctions of the literals. We say such a rule is *basic*. Also, we call a rule that has more complicated condition as *non-basic* rule. A non-basic rule must be transformed into multiple basic rules, whose conditions are disjointed with each other. For example, a non-basic rule that has form $(a \vee b \vee c) \to v$, is transformed into three basic rules, i.e., $(a) \to v$, $(\neg a \wedge b) \to v$, and $(\neg a \wedge \neg b \wedge c) \to v$.

Ohta *et al.* [10] identified a relation between two rules that can be classified into four non-overlapping and exhaustive cases. Based on this classification, they identified the condition of a set of rules as consistent, i.e., there exists at least one coalition structure CS such that each rule is applicable to a coalition in CS. Furthermore, they develop mixed integer programming (MIP) formulations for finding a consistent rule set that maximizes the sum of the rule values.

[1]For example, if a rule has a form $\neg a_1 \to 1$ and there exist agents $a_1, a_2, \ldots, a_n$, we can create equivalent rules as follows: $\neg a_1 \wedge a_2 \to 1$, $\neg a_1 \wedge \neg a_2 \wedge a_3 \to 1$, $\ldots$, $\neg a_1 \wedge \neg a_2 \ldots \wedge \neg a_{n-1} \wedge a_n \to 1$.

2.2 Partition Function Game

When there exist externalities among coalitions, the value of a coalition depends on the coalition structure in which the coalition belongs. An *embedded coalition* is a pair (S, CS), where $S \in CS \in \Pi(A)$. Let us denote the set of all embedded coalitions as M, i.e., $M := \{(S, CS) : CS \in \Pi(A), S \in CS\}$. A partition function is a mapping $w : M \to \mathbb{R}$.

Michalak *et al.* [8] proposed a concise representation of a partition function called *embedded MC-nets*.

Definition 2 (Embedded MC-nets). *An* embedded MC-nets *consists of set of embedded rules ER. Each embedded rule $er \in ER$ is of the form: $(L_1)|(L_2), \ldots, (L_l) \to v_{er}$, where each $L_1, L_2, \ldots, L_l$ is the conjunctions of literals over A. L_1, which we call the internal condition, is the condition that must be satisfied in the coalition that receives the value. $L_2, \ldots, L_l$, which we call external conditions, must be satisfied in other coalitions. We say that an embedded rule er is* applicable *to coalition S in CS if L_1 is applicable to S and each of $L_2, \ldots, L_l$ is applicable to some coalition $S' \in CS \setminus \{S\}$. For coalition S, $w(S, CS)$ is given as $\sum_{er \in ER_{(S,CS)}} v_{er}$, where $ER_{(S,CS)}$ is the set of embedded rules applicable to S in CS.*

Note that for an embedded rule, there exists an implicit constraint such that external conditions must be satisfied in coalitions $CS \setminus \{S\}$. By adding each positive literal in internal condition L_1 to the negative literals of each external conditions $L_2, \ldots, L_l$, as well as by adding each positive literal in external conditions $L_2, \ldots, L_l$ to the negative literals of internal condition L_1, we can explicitly represent this implicit constraint. We say an embedded rule is in an *explicit form* if the above condition is satisfied. For example, if an original rule is $(a)|(b), (c) \to v$, its explicit form is $(a \wedge \neg b \wedge \neg c)|(b \wedge \neg a), (c \wedge \neg a) \to v$. For simplicity, in the rest of this paper, we assume each embedded rule is in an explicit form.

Example 1. *Let us assume the following rules. Here, er_1 is an embedded rule.*

$r_1 : (a) \to 1,$	$r_2 : (b) \to 1,$
$r_3 : (c) \to 1,$	$r_4 : (d \wedge \neg a \wedge \neg b) \to 3,$
$r_5 : (a \wedge b) \to 1,$	$er_1 : (d \wedge \neg a \wedge \neg b)|(a \wedge b \wedge \neg d) \to -2.$

If $CS = \{\{ab\}, \{c\}, \{d\}\}$, all rules are applicable. Thus, the $V(CS) = 1 + 1 + 1 + 3 + 1 - 2 = 5$.

The representation of embedded MC-nets is fully expressive and at least as concise as the conventional partition function game representation. As far as the authors are aware, the problem of finding an optimal coalition structure when a game is represented by embedded MC-nets has not yet been investigated. Extending the MIP formulation in [10] to handle embedded MC-nets is rather straightforward, as long as the rule has a non-negative value. More specifically, for an embedded rule that has a form $er : (L_1)|(L_2), \ldots, \to v_{er}$, we create basic rules $r_1 : (L_1) \to 0$, $r_2 : (L_2) \to 0$, $\ldots$, $r_l : (L_l) \to 0$. Assume $x_{er}, x_{r_1}, \ldots, x_{r_l}$ are 0/1 decision variables in the MIP formulation, i.e., when the value is 1, the rule is selected. An objective function is given by $\sum_{er} v_{er} \cdot x_{er}$. Also, we add a constraint that x_{er} can be 1 only when all of $x_{r_1}, \ldots, x_{r_l}$ are 1. Note that such a constraint is not linear. However, there exists a well-known encoding trick to represent such a non-linear constraint in MIP formulations [3].

3. CSG WITH NEGATIVE VALUE RULES

Although any characteristic function can be represented without using negative value rules, this restriction can make the representation size of a problem exponentially large. For example, let us assume for set of agents $A = \{a_1, a_2, \ldots, a_n\}$, $v(S) = |S|$ if $S \neq A$ and $v(A) = 0$. If we naively represent this characteristic function without negative value rules, we need $2^n - 1$ rules, where each rule is applicable exactly to one coalition[2]. If we can use a negative value rule, it suffices to have $n + 1$ rules, i.e., for each $a_i \in A$, $(a_i) \to 1$, and one negative value rule $(a_1 \wedge a_2 \wedge \ldots \wedge a_n) \to -n$.

However, handling negative value rules is a challenging task. In general, a negative reward in a reward maximization problem, or a negative cost in a cost minimization problem, is considered as a nuisance. When all rules have positive values, choosing a rule never hurts. Thus, in the MIP formulation, we construct a solver so that it tries to choose as many rules as possible. The constraints only specify the conditions where rules cannot be selected at the same time. Thus, if we simply include a negative value rule, the solver just ignores this rule, since choosing it *hurts*. We must describe the condition where the solver is forced to choose this negative value rule as a result of choosing other positive rules. Such a condition involves interaction among multiple rules, which is difficult to handle efficiently.

4. FULL TRANSFORMATION

Since handling negative value rules is difficult for the MIP formulation in [10], we consider transforming a rule set, which contains both positive/negative value rules into a rule set that contains positive value rules only.

4.1 MC-nets

We first show a full transformation algorithm for MC-nets. We assume that R is divided into two groups, i.e., a set of positive value rules R_+ and a set of negative value rules R_-.

Definition 3 (Full transformation algorithm). *The full transformation algorithm is defined as follows.*

1. *Set $R'_- = R_-$, $R'_+ = R_+$.*
2. *If $R'_- = \emptyset$, return R'_+.*
3. *Remove one rule $r_x : (L_x) \to -v_x$ from R'_-.*
4. *Remove one rule $r_i : (L_i) \to v_i$ from R'_+, such that $L_x \wedge L_i \not\models \bot$. If no such rule exists, return failure.*
5. *If $\neg L_x \wedge L_i \not\models \bot$, create a set of basic rules that is the transformation of non-basic rule $(\neg L_x \wedge L_i) \to v_i$. Add them to R'_+.*
6. *Create new basic rule $(L_x \wedge L_i) \to v_i - v_x$. If $v_i - v_x > 0$, add this rule to R'_+. If $v_i - v_x < 0$, add it to R'_-.*
7. *If $L_x \wedge \neg L_i \not\models \bot$, create a set of basic rules that is the transformation of non-basic rule $(L_x \wedge \neg L_i) \to -v_x$. Add them to R'_-. Goto 2.*

Let us explain the basic ideas of this algorithm. Since we assume that $\forall S, v(S) \geq 0$ holds, if negative value rule $r_x : (L_x) \to -v_x$ is applicable to coalition S, there exists at least one positive value rule $r_i : (L_i) \to v_i$, which is also applicable to S. In other words, r_i can partially eliminate the effect of r_x. We transform r_x and r_i into the following three rules:

[2]If we use a clever encoding trick, we require $O(n^2)$ rules.

r'_1: $(\neg L_x \wedge L_i) \rightarrow v_i$, which is added in Step 5,

r'_2: $(L_x \wedge L_i) \rightarrow v_i - v_x$, which is added in Step 6, and

r'_3: $(L_x \wedge \neg L_i) \rightarrow -v_x$, which is added in Step 7.

It is obvious that the original two rules, r_x and r_i, and these three rules are equivalent. Since r'_1 and r'_3 are non-basic, they must be transformed into multiple basic rules.

We can guarantee that the full transformation algorithm terminates, i.e., the following theorem holds.

THEOREM 1. *The full transformation algorithm terminates.*

PROOF. By one iteration of this algorithm, the negative value rule r_x is eliminated if $L_x \wedge \neg L_i \models \perp$ and $v_i \geq v_x$. If $L_x \wedge \neg L_i \not\models \perp$, a set of negative value rules are added in Step 7, but the conditions of these rules, i.e., $L_x \wedge \neg L_i$, are more specific than L_x. Also, if $v_i < v_x$, a new negative value rule is added in Step 6, but the condition of this rule, i.e., $L_x \wedge L_i$ is more specific than L_x and also disjoint with $L_x \wedge \neg L_i$. Furthermore, the value of this rule, i.e., $v_i - v_x$ is closer to 0 than the original value $-v_x$. Thus, by one iteration of this algorithm, the conditions of negative value rules become more specific and/or the negative value becomes closer to 0. Therefore, this algorithm cannot iterate infinitely and will terminate eventually. □

EXAMPLE 2. *Let us describe the full transformation algorithm, assuming $r_x : (L_x) \rightarrow -1$, where $L_x = a \wedge d \wedge e$, and $r_1 : (L_1) \rightarrow 1$, where $L_1 = a \wedge \neg b \wedge \neg c$, are selected. Since $L_1 \wedge \neg L_x = (a \wedge \neg b \wedge \neg c) \wedge (\neg a \vee \neg d \vee \neg e) \not\models \perp$ holds, we create non-basic rule $(L_1 \wedge \neg L_x) \rightarrow 1$ in Step 5. Then, we obtain two basic rules from this rule: $(a \wedge \neg b \wedge \neg c \wedge \neg d) \rightarrow 1$ and $(a \wedge \neg b \wedge \neg c \wedge d \wedge \neg e) \rightarrow 1$. We do not create any new rule in Step 6 since $v_{r_1} + v_{r_x} = 1 - 1 = 0$. Finally, since $\neg L_1 \wedge L_x = (\neg a \vee b \vee c) \wedge (a \wedge d \wedge e) \not\models \perp$ holds, we create non-basic rule $(\neg L_1 \wedge L_x) \rightarrow -1$. Then, we obtain two basic rules from this rule: $(a \wedge b \wedge d \wedge e) \rightarrow -1$ and $(a \wedge \neg b \wedge c \wedge d \wedge e) \rightarrow -1$.*

By using this algorithm, we can eliminate all negative value rules. However, this approach is not scalable. There exists an instance where the number of newly generated rules becomes $\Omega(n^2)$ by using the full transformation algorithm.

EXAMPLE 3. *Let us consider the following rules.*

r_0: $(p_0 \wedge \neg n_1 \wedge \neg n_2 \wedge \ldots \wedge \neg n_k) \rightarrow 1$

r_1: $(p_1 \wedge n_1) \rightarrow 1$

r_2: $(p_2 \wedge n_2) \rightarrow 1$

...

r_k: $(p_k \wedge n_k) \rightarrow 1$

r_x: $(p_0 \wedge p_1 \wedge p_2 \wedge \ldots \wedge p_k) \rightarrow -1$

This rule set contains $k+1$ positive value rules and one negative value rule, where the total number of agents is $2k+1$. Figure 1 shows the number of newly generated rules from this rule sets by varying k. We can see that the number of newly generated rules becomes $\Omega(k^2)$, which is also $\Omega(n^2)$.

Then, can we reduce the number of required rules by using more clever encoding trick? Actually, the answer is no, i.e., the following theorem holds.

THEOREM 2. *To represent the characteristic function in Example 3 by using positive value rules only, we need $\Omega(n^2)$ rules.*

PROOF. For all $1 \leq i < j \leq k$, we denote $\{p_0, p_1, \ldots, p_k, n_i, n_j\}$ as $S_{i,j}$. For $S_{i,j}$, only rules r_x, r_i, r_j are applicable, thus $v(S_{i,j})$ is equal to 1. Assume that a set of positive value rules R'_+ represents v. There must be at least one rule in R'_+ that is applicable to $S_{i,j}$. Let us represent such a rule as $r_{i,j}$.

Now, we show that $r_{i,j}$ is not applicable to any $S_{i',j'}$, where $1 \leq i' < j' \leq k$ and $i \neq i' \vee j \neq j'$. We derive a contradiction by assuming that $r_{i,j}$ is applicable to $S_{i',j'}$.

When $i = i'$ or $i = j'$, let us consider coalition $S = \{p_0, p_1, \ldots, p_k, n_i\}$. For S, only rules r_x, r_i are applicable, thus $v(S)$ is equal to 0. However, we show that $r_{i,j}$ is applicable to S, thus $v(S)$ cannot be 0. $r_{i,j}$ is not applicable to S, if (i) its positive literals include agent n_l, where $l \neq i$, or (ii) its negative literals include at least one of $\{p_0, p_1, \ldots, p_k, n_i\}$. For (i), if $l = j$, $r_{i,j}$ is not applicable to $S_{i',j'}$. Also, if $l \neq j$, $r_{i,j}$ is not applicable to $S_{i,j}$. For (ii), $r_{i,j}$ is not applicable to both of $S_{i,j}$ and $S_{i',j'}$. This contradicts the assumption that $r_{i,j}$ is applicable to both of $S_{i,j}$ and $S_{i',j'}$. We can use a similar argument for the cases where $j = i'$ or $j = j'$.

Then, let us consider the case that i, j, i', j' are different with each other. Let us consider coalition $S = \{p_0, p_1, \ldots, p_k\}$. For S, only rules r_x, r_0 are applicable, thus $v(S)$ is equal to 0. However, we show that $r_{i,j}$ is applicable to S, thus $v(S)$ cannot be 0. $r_{i,j}$ is not applicable to S, if (i) its positive literals include agent n_l, where $1 \leq l \leq k$, or (ii) its negative literals include at least one of $\{p_0, p_1, \ldots, p_k\}$. For (i), if $l = i$ or $l = j$, $r_{i,j}$ is not applicable to $S_{i',j'}$. If $l \neq i$ and $l \neq j$, $r_{i,j}$ is not applicable to $S_{i,j}$. For (ii), $r_{i,j}$ is not applicable to both of $S_{i,j}$ and $S_{i',j'}$. This contradicts the assumption that $r_{i,j}$ is applicable to both of $S_{i,j}$ and $S_{i',j'}$.

Thus, for each i, j, where $1 \leq i < j \leq k$, there must be distinct element $r_{i,j}$ in R'_+, and the number of elements in R'_+ must be at least $k(k-1)/2$, which is $\Omega(n^2)$. □

It remains an open question whether there exists a characteristic function and MC-nets representation with negative value rules, such that representing this characteristic function by a MC-net without negative value rules requires exponentially more space compared to the original MC-net representation. Our current conjecture is that such a characteristic function is likely to exist in embedded MC-nets, but not in MC-nets, assuming the number of negative value rules is bounded. We will discuss this issue in the next subsection.

4.2 Embedded MC-nets

The full transformation algorithm presented in Section 4.1 can be easily extended to embedded MC-nets. We replace a condition such as L_i to the condition for embedded rule C_{er}, which is a pair of internal condition L_1 and external conditions $L_2, \ldots, L_l$.

One tricky point is how to create the negation of C_{er}. Recall that embedded rule er is applicable to coalition S in CS if L_1 is applicable to S and each of $L_2, \ldots, L_l$ is applicable to some coalition $S' \in CS \setminus \{S\}$. Thus, er is not applicable to coalition S in CS if (i) L_1 is not applicable to S, (ii) L_1 is applicable to S, but L_2 is not applicable to *any* coalition in $CS \setminus \{S\}$, (iii) L_1 is applicable to S and L_2 is applicable to some coalition $S' \in CS \setminus \{S\}$, but L_3 is not applicable to *any* coalition in $CS \setminus \{S\}$, and so on. Handling case (i) is easy. Let us examine how to handle case (ii). Assume $L_2 = p_1 \wedge p_2$. We must guarantee that for any coalition $S' \in CS \setminus \{S\}$, $\neg L_2 = \neg p_1 \vee \neg p_2$ holds. If S' does not contain p_1, $\neg L_2$ holds. If S' contains p_1, then S' must satisfy $\neg p_2$. Since there exists exactly one coalition that contains p_1, it is sufficient to guarantee that there exists some coalition $S' \in CS \setminus \{S\}$, such that $p_1 \wedge \neg p_2$ holds.

To summarize, in order to represent $\neg C_{er}$, where C_{er} is a pair of the internal condition L_0 and external conditions $L_1, \ldots, L_l$, we need following conditions (here, we assume each $L_i = l_{i_1} \wedge l_{i_2} \wedge l_{i_3} \wedge \ldots$): (i) $(\neg L_0)$, (ii) $(L_0)|(l_{1_1} \wedge \neg l_{1_2})$, $(L_0)|(l_{1_1} \wedge l_{1_2} \wedge \neg l_{1_3})$, ..., (iii) $(L_0)|(L_1)(l_{2_1} \wedge \neg l_{2_2})$, $(L_0)|(L_1)(l_{2_1} \wedge l_{2_2} \wedge \neg l_{2_3})$, and so on.

EXAMPLE 4. *Let us consider the following rules:*

r_x: $(a \wedge \neg p_1 \wedge \neg p_2 \ldots \wedge \neg p_{k+1})$ |
$(\neg a \wedge p_1 \wedge \neg p_2 \wedge \neg p_3 \ldots \wedge \neg p_{k+1})$,
$(\neg a \wedge p_2 \wedge \neg p_1 \wedge \neg p_3 \ldots \wedge \neg p_{k+1}), \ldots,$
$(\neg a \wedge p_{k+1} \wedge \neg p_1 \wedge \neg p_2 \ldots \wedge \neg p_k) \rightarrow -1$,

r_1: $(a \wedge \neg p_1 \wedge \neg p_2 \ldots \wedge \neg p_{k+1})$ |
$(\neg a \wedge p_1 \wedge \neg p_2 \wedge \neg p_3 \ldots \wedge \neg p_{k+1} \wedge \neg h_1 \wedge \neg h_2 \ldots \wedge \neg h_k)$,
$(\neg a \wedge p_2 \wedge \neg p_1 \wedge \neg p_3 \ldots \wedge \neg p_{k+1})$,
$\ldots,$
$(\neg a \wedge p_{k+1} \wedge \neg p_1 \wedge \neg p_2 \ldots \wedge \neg p_k) \rightarrow 1$,

...

r_{k+1}: $(a \wedge \neg p_1 \wedge \neg p_2 \ldots \wedge \neg p_{k+1})$ |
$(\neg a \wedge p_1 \wedge \neg p_2 \wedge \neg p_3 \ldots \wedge \neg p_{k+1})$,
$(\neg a \wedge p_2 \wedge \neg p_1 \wedge \neg p_3 \ldots \wedge \neg p_{k+1})$,
$\ldots,$
$(\neg a \wedge p_{k+1} \wedge \neg p_1 \wedge \neg p_2 \ldots \wedge \neg p_k \wedge \neg h_1 \wedge \neg h_2 \ldots \wedge \neg h_k) \rightarrow 1$

This rule set contains $k+1$ positive value rules and one negative value rule. The total number of agents is $2k+2$. This rule set is constructed based on the well-known pigeon hole principle. There are $k+1$ pigeons $p_1, \ldots, p_{k+1}$ and k holes $h_1, \ldots, h_k$. r_x requires that all pigeons are in different coalitions. Also, each r_i is true if no hole is assigned to pigeon p_i. Thus, $\neg r_i$ means pigeon p_i has (at least) one hole. Then, as long as r_x is true, at least one rule $r_1, \ldots, r_k$ must be true. Otherwise, each pigeon has a different hole to stay, which is clearly impossible since there exist only k holes while there exist $k+1$ pigeons.

Figure 2 shows the number of newly generated rules (which includes embedded rules) by using our transformation algorithm. We can see that the number of newly generated rules becomes $\Omega(2^k)$, thus it is $\Omega(2^n)$. Although we have not yet proved that this exponential blowup is really inevitable, our current conjecture suggest that it is actually inevitable. Even if this is not the case, the current results are already very negative. Since the CSG algorithm presented in [10] is exponential in the number of rules, even the increase of $\Theta(n^2)$ can be problematic.

5. PARTIAL TRANSFORMATION

In this section, we develop an alternative transformation algorithm, which does not eliminate all negative value rules. More specifically, we are going to transform rules that contain negative value rules into positive value rules and some negative value rules that have a special form defined as follows.

DEFINITION 4 (SINGLETON RULE). *We say rule $r : (L) \rightarrow v_r$ is singleton if L consists of exactly one positive literal a.*

It is clear that such a singleton rule r is applicable to any CS, since agent a must belong to one coalition. Also, for any other rule r', choosing r never prohibits choosing r'. Thus, if CS^* is optimal without r, then it is optimal with r. Thus, when solving the optimization problem, we can just ignore r. After finding optimal CS^*, we adjust $V(CS^*)$ by adding v_r.

Thus, if a negative value rule is singleton, it is actually harmless. Our new partial transformation algorithm transforms all negative value rules into positive value rules and negative value singleton rules.

DEFINITION 5 (PARTIAL TRANSFORMATION ALGORITHM). *The partial transformation algorithm is defined as follows.*

1. *Set $R'_- = R_-$, $R'_+ = R_+, R_{\text{singleton}} = \emptyset$.*
2. *If $R'_- = \emptyset$, return R'_+ and $R_{\text{singleton}}$.*
3. *Remove one rule $r_x : (L_x) \rightarrow -v_x$ from R'_-.*
4. *Choose one positive literal $a \in L_x$. Add one rule $r_{\text{singleton}}$: $(a) \rightarrow -v_x$ to $R_{\text{singleton}}$.*
5. *Create a set of basic rules that are the transformation of non-basic rule $(\neg L_x \wedge a) \rightarrow v_x$. Add them to R'_+. Goto 2.*

Let us describe the procedure of this algorithm. In Step 4, we add one negative value singleton rule $r_{\text{singleton}} : (a) \rightarrow -v_x$. Conceptually, we also add one positive value singleton rule $r_+ : (a) \rightarrow v_x$. It is clear that adding these two rules does not change the values of a characteristic function since they negate with each other. Note that the procedure in Step 5 is equivalent to the full transformation algorithm in Steps 5 to 7, assuming that we choose r_+ as a positive value rule. No rule is added since $v_x - v_x = 0$ holds in Step 6, and $L_x \wedge \neg a \models \perp$ holds in Step 7. We iterate this procedure until all negative non-singleton value rules are eliminated. Then, we have a new rule set, which contains only positive value rules R'_+ and negative value singleton rules $R_{\text{singleton}}$.

As described above, we can solve CSG by using only R'_+ and obtain CS^*. The true value of $V(CS^*)$ is obtained by adding the values of the singleton rules in $R_{\text{singleton}}$.

THEOREM 3. *The partial transformation algorithm terminates.*

PROOF. By one iteration of this algorithm, one negative value rule is eliminated and no negative literal is added. Therefore, this algorithm terminates after $|R_-|$ iterations. □

THEOREM 4. *In MC-nets, each negative value rule is transformed into one singleton negative value rule and $m-1$ positive value rules, where m is the maximal number of agents included in the rule.*

PROOF. Let us assume r_x is the form $(L_x) \rightarrow -v_x$, where $L_x = (a_1 \wedge a_2 \wedge \cdots \wedge a_k \wedge \neg a_{k+1} \wedge \cdots \wedge \neg a_m)$. We first create non-basic rule $\neg L_x \wedge a_1 \rightarrow v_x$ in Step 5. Then, it is transformed into $m-1$ basic rules i.e., $(a_1 \wedge \neg a_2) \rightarrow v_x$, $(a_1 \wedge a_2 \wedge \neg a_3) \rightarrow v_x$, ..., $(a_1 \wedge a_2 \wedge \cdots \wedge \neg a_k) \rightarrow v_x$, $(a_1 \wedge a_2 \wedge \cdots \wedge a_k \wedge a_{k+1}) \rightarrow v_x$, ..., $(a_1 \wedge \cdots \wedge a_k \wedge \neg a_{k+1} \wedge \cdots \wedge \neg a_{m-1} \wedge a_m) \rightarrow v_x$. Thus, each negative value rule is transformed into one singleton negative value rule and $m-1$ positive value rules. □

Theorem 5. *In embedded MC-nets, each negative value rule is transformed into one singleton negative value rule and $O(m \cdot l)$ positive value rules, where l is the maximal number of basic rules included in the embedded rule, and m is the maximal number of agents included in each basic rule.*

Proof. Let us assume er_x has a form $(C_{er_x}) \to -v_x$, where $C_{er_x} = (L_0)|(L_1)\dots(L_{l-1})$. Also, each L_i has a form $(p_{i_1} \wedge p_{i_2} \wedge \dots \wedge p_{i_s} \wedge \neg n_{i_1} \wedge \neg n_{i_2} \wedge \dots \wedge \neg n_{i_t})$, where $s + t \le m$. We first create non-basic rule $\neg C_{er_x} \wedge p_{0_i} \to v_x$ in Step 5. In order to represent $\neg C_{er_x}$, we need following conditions: (i) $(\neg L_0)|\emptyset$, (ii) $(L_0)|(p_{1_1} \wedge \neg p_{1_2})$, $(L_0)|(p_{1_1} \wedge p_{1_2} \wedge \neg p_{1_3})$, ..., $(L_0)|(p_{1_1} \wedge p_{1_2} \wedge \dots p_{1_s} \wedge n_{1_1})$, ..., $(L_0)|(p_{1_1} \wedge p_{1_2} \wedge \dots p_{1_s} \wedge \neg n_{1_1} \wedge \dots n_{1_t})$, (iii) $(L_0)|(L_1)(p_{2_1} \wedge \neg p_{2_2})$, $(L_0)|(L_1)(p_{2_1} \wedge p_{2_2} \wedge \neg p_{2_3})$, ..., $(L_0)|(L_1)(p_{2_1} \wedge p_{2_2} \wedge \dots p_{2_s} \wedge n_{2_1})$, ..., $(L_0)|(L_1)(p_{2_1} \wedge p_{2_2} \wedge \dots p_{2_s} \wedge \neg n_{2_1} \wedge \dots n_{2_t})$, (iv) $(L_0)|(L_1)(L_2)(p_{3_1} \wedge \neg p_{3_2})$, ..., and so on. Thus, we create at most m basic rules from each part of $\neg C_{er_x}$. Since there exist l parts, each negative value rule is transformed into one singleton negative value rule and $O(m \cdot l)$ positive value embedded rules. □

Example 5. *Let us consider a negative value embedded rule that has the following form: $r_x : (L_1)|(L_2), (L_3), \dots, (L_n) \to -1$, where each $L_i = a_i \wedge \bigwedge_{j \ne i} \neg a_j$.*

In other words, this rule means each agent creates its own coalition. This rule contains n basic rules, and each basic rule contains n agents. Thus, this rule gives the worst-case where the number of rules added in the partial transformation algorithm is maximized, i.e., it becomes $\Theta(n^2)$. To make matters worse, most of these rules are embedded rules. Each embedded rule contains at most n basic rules. Thus, in this worst case, the total increase of the basic rules becomes $\Theta(n^3)$. In general, the partial transformation algorithm creates $O(m \cdot l^2)$ basic rules.

6. DIRECT ENCODING

In this section, we develop another approach for handling negative value rules. Instead of transforming a negative value rule into positive value rules, we add several dummy rules, whose rule values are zero. Each dummy rule describes some situations where the negative value rule is inapplicable. Furthermore, we add a constraint for the optimization algorithm so that the negative value rule must be selected if all of the dummy rules are not selected.

Definition 6 (Dummy rules (for basic rule)). *Assume there exists a negative value rule $r_x : (L_x) \to -v_x$ $(v_x > 0)$, where $L_x = a_1 \wedge a_2 \wedge \dots \wedge a_k \wedge \neg a_{k+1} \wedge \neg a_{k+2} \wedge \dots \wedge \neg a_m$. Dummy rules generated by this negative value rule are of the following two types:*

(i) $(a_1 \wedge \neg a_i) \to 0$, where $2 \le i \le k$,

(ii) $(a_1 \wedge a_j) \to 0$, where $k+1 \le j \le m$.

We denote $D(L_x)$ as a set of dummy rules created from L_x.

It is obvious that the following theorem holds.

Theorem 6. *For each negative value rule, $m-1$ dummy rules are created, where m is the maximal number of agents included in the rule.*

Also, the following theorem holds.

Theorem 7. *A negative value rule is applicable to a coalition in coalition structure CS if and only if all of its dummy rules are not applicable to any coalition in CS.*

Proof. The condition of a dummy rule can be either $a_1 \wedge \neg a_i$ or $a_1 \wedge a_j$. In either case, it is clear that when this dummy rule is applicable to one coalition in CS, the negative value rule is not applicable to any coalition in CS. Also, if all dummy rules are inapplicable to any coalition in CS, it means that $a_1, a_2, \dots, a_k$ are in identical coalition S, while $a_{k+1}, a_{k+2}, \dots, a_m$ are not in S. Thus, the negative value rule is applicable to S. □

Assume $x_{r_x}, x_{d_1}, \dots$ are 0/1 decision variables for negative value rule r_x and its dummy rules $D(L_x)$ in the MIP formulation. We add a constraint that at least one of $x_{r_x}, x_{d_1}, \dots$ must be 1, i.e., $x_{r_x} + x_{d_1} + \dots \ge 1$ holds.

Let us show an example of dummy rules. We create the following dummy rules $D(L_x)$ for negative value rule r_x presented in Example 3: $(p_0 \wedge \neg p_1) \to 0$, $(p_0 \wedge \neg p_2) \to 0$, ..., $(p_0 \wedge \neg p_k) \to 0$. These dummy rules $D(L_x)$ are quite similar to the rules added in the partial transformation algorithm, which we denote $R(L_x) = \{(p_0 \wedge \neg p_1) \to 1, (p_0 \wedge p_1 \wedge \neg p_2) \to 1, \dots, (p_0 \wedge p_1 \wedge \dots \wedge p_{k-1} \wedge \neg p_k) \to 1\}$. Actually, both algorithms add the same number of rules. However, dummy rules are much simpler. This is because the rules in $R(L_x)$ must be disjoint with each other, while the dummy rules can overlap (since their rule values are zero).

Definition 7 (Dummy rules (for embedded rule)). *Assume there exists a negative value embedded rule $r_x : (L_1)|(L_2), \dots, (L_l) \to -v_x$ $(v_x > 0)$. Then, dummy rules for r_x is $\bigcup_{L_i} D(L_i)$.*

It is obvious that the following theorem holds.

Theorem 8. *For each negative value embedded rule, one singleton negative value rule and $(m-1) \cdot l$ dummy rules are created.*

Also, the following theorem holds.

Theorem 9. *A negative value embedded rule is applicable to a coalition with coalition structure CS if and only if all of its dummy rules are not applicable to any coalition in CS*

We omit the proof since it is basically identical to Theorem 7.

For example, we create the following dummy rules for negative value embedded rule r_x presented in Example 5: $(a_1 \wedge a_2) \to 0$, ..., $(a_1 \wedge a_n) \to 0$, $(a_2 \wedge a_3) \to 0$, ..., $(a_2 \wedge a_n) \to 0$, ..., $(a_{n-1} \wedge a_n) \to 0$.

The number of these dummy rules is about the same as the number of rules added in the partial transformation algorithm. However, each dummy rule is a simple basic rule. Thus, in the worst case, the total increase of basic rules in the direct encoding approach is $\Theta(n^2)$, while it is $\Theta(n^3)$ in the partial transformation algorithm.

7. EVALUATIONS

We experimentally evaluated the performance of proposed methods. All of the tests were run on a Xeon E5540 processor with 12-GB RAM. The test machine runs Windows Vista Business x64 Edition SP2. We used CPLEX 12.1, a general-purpose mixed integer programming package.

We show the performance of our proposed algorithms with randomly generated instances for the following two cases:

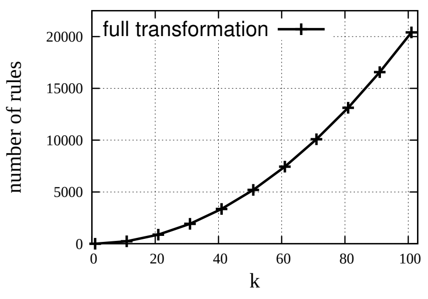


Figure 1: # of Generated Rules (Example 3)

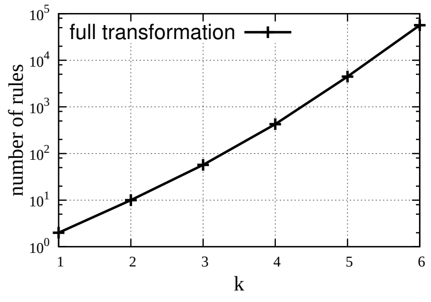


Figure 2: # of Generated Rules (Example 4)

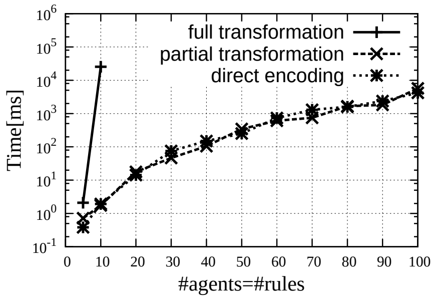


Figure 3: Computation Time: Case (i)

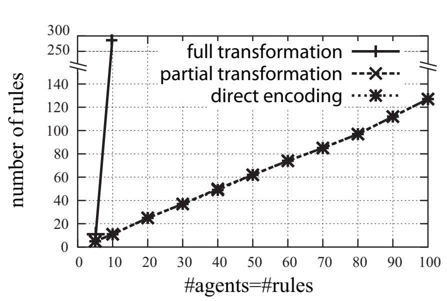


Figure 4: # of Generated Rules: Case (i)

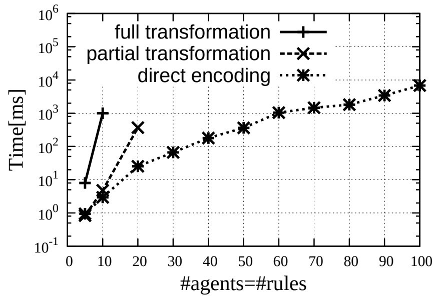


Figure 5: Computation Time: Case (ii)

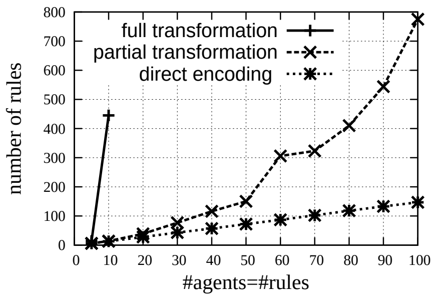


Figure 6: # of Generated Rules: Case (ii)

(i) rules can be either positive/negative and non-embedded, and (ii) rules can be either positive/negative and either non-embedded/embedded. For each case, problem instances are generated in the following method.

For case (i), we use a decay distribution [14] described as follows. Create a coalition with one random agent. Then repeatedly add a new random agent with probability α until an agent is not added or the coalition includes all agents. We use $\alpha = 0.55$. Choose value $v(S)$ between 0 and the number of agents in S uniformly at random. Then we create rule $(\bigwedge_{a \in S} a) \rightarrow v(S)$. Furthermore, we modify each rule by randomly moving an agent from the positive to negative literals with probability p. We use $p = 0.2$. Also, we convert the value to be negative with probability q. This method is basically identical to that used in [10].

For case (ii), we first create rule $(L_1) \rightarrow v_{er}$ in the same way as described in (i). Then, we repeatedly add a new condition of a rule with probability β until a rule is not added. We use $\beta = 0.15$. The value of the generated embedded rule is chosen between 0 and the number of agents in the rule, uniformly at random. Then we convert it to be negative with probability $q = 0.2$.

For all generated problem instances, we make sure that no coalition has a negative value. More specifically, we add positive value rules, if some coalition has a negative value.

We limit the time for problem transformation to five minutes, i.e., if an algorithm fails to transform a problem instance within five minutes, we terminate the transformation and do not examine the computation time for such an instance. Such an early termination occurs frequently in the full transformation approach, but it never happens in the partial transformation and direct encoding algorithms.

We set #rules = #agents and vary #rules from 5 to 100. We generate 50 problem instances for all cases and #rules. We investigate the computation time for these three algorithms and the number of newly generated rules. The results are illustrated in Figures 3, 4, 5, and 6. Each data point in these Figures is the median of 50 data points.

The results of case (i) are illustrated in Figures 3 and 4. We can see that the CSG problem becomes more difficult when #rules (which is equal to #agents) increases in Figure 3. This is natural since #rules increases, the number of decision variables in the corresponding MIP formulation.

It is clear that the full transformation algorithm is inefficient compared with the other two algorithms. It cannot transform problem instances in five minutes even when #rules = 15. Also, we were able to solve only 45% of the problem instances; the remaining 55% problem instances cause error due to insufficient memory. On the other hand, the partial transformation and direct encoding algorithms are more efficient than the full transformation algorithm. The required time for these algorithms is less than 10 seconds. We can see that the number of the newly generated rules of these two algorithms is almost the same in Figure 4.

The differences of these algorithms become more obvious in Figures 5 and 6. The full transformation algorithm cannot transform the problem instances in five minutes when #rules = 15. Also, we were able to solve only 45% of the problem instances; the remaining 55% problem instances cause error due to insufficient memory. The partial transformation algorithm can solve problem instances within 1.0 seconds when #rules = 20, but it fails to solve problem instances when #rules becomes more than 30 due to insufficient memory. On the other hand, the direct encoding algorithm can solve problem instances for all #rules and the required time is less than 10 seconds. Figure 6 shows the number of newly generated basic rules, i.e., a newly generated embedded rule is further decomposed into multiple basic rules. The partial transformation algorithm adds $O(m \cdot l)$ embedded rules, each of which contains at most l basic rules. Thus, the number of basic rules is $O(m \cdot l^2)$. On the other hand, the direct encoding algorithm adds at most $O(m \cdot l)$ basic rules. Consequently, the problem instances generated by the partial transformation algorithm becomes more difficult than those of the direct encoding algorithm.

To summarize, the direct encoding algorithm is the most scalable among these three algorithms; the required times for the solving problem instances in all cases are all less than 10 seconds. On the other hand, the $\text{IP}^{+/-}$ algorithm, which does not make use of compact representations, required around 160 minutes to solve instance with 20 agents [12].

8. CONCLUSIONS

In this paper, we extended the formalization of CSG in [10] so that it can handle negative value rules. Allowing negative value rules is important since (a) it can reduce the efforts for describing a characteristic function, (b) the representation size of a problem can be much more concise, and (c) in many realistic situations, it is natural to assume that a coalition has negative externalities to other coalitions.

Since the current CSG algorithm cannot handle negative value rules, we examine the following three methods: (i) a full transformation algorithm, which transforms all negative value rules into positive value rules, (ii) a partial transformation algorithm, which transforms all negative value rules into positive value rules and some negative rules that have a special form, and (iii) a direct encoding algorithm, which creates a set of *dummy rules* so that negative value rules are handled appropriately.

We show that the full transformation algorithm is not scalable in MC-nets since the worst-case representation size will be $\Omega(n^2)$ for MC-nets, and it can be $\Omega(2^n)$ for embedded MC-nets. On the other hand, by using the partial transformation or direct encoding algorithms, an exponential blow-up never occurs even for embedded MC-nets. We experimentally compared these algorithms and showed that the direct encoding algorithm is by far superior.

It still remains an open question whether a characteristic function exists that inevitably requires exponentially more space without using negative value rules. In our future works, we hope to confirm our current conjectures, i.e., such a characteristic function exists in embedded MC-nets (one promising candidate is the characteristic function presented in Example 4), but not in MC-nets.

9. ACKNOWLEDGMENTS

This work was supported by KAKENHI (20240015) and Grant-in-Aid for JSPS research fellows. The authors would like to thank Hirotaka Ono, Ryuzo Hasegawa, Hiroshi Fujita, Miyuki Koshimura and anonymous reviewers for their helpful comments.

10. REFERENCES

[1] H. Aziz and B. de Keijzer. Complexity of coalition structure generation. In *AAMAS*, pages 191–198, 2011.

[2] B. Banerjee and L. Kraemer. Coalition structure generation in multi-agent systems with mixed externalities. In *AAMAS*, pages 175–182, 2010.

[3] E. Castillo, A. J. Conejo, P. Pedregal, R. Garcia, and N. Alguacil. *Building and Solving Mathematical Programming Models in Engineering and Science.* Wiley-Interscience, 2001.

[4] V. Conitzer and T. Sandholm. Complexity of constructing solutions in the core based on synergies among coalitions. *Artificial Intelligence*, 170(6):607–619, 2006.

[5] E. Elkind, L. A. Goldberg, P. W. Goldberg, and M. Wooldridge. A tractable and expressive class of marginal contribution nets and its applications. In *AAMAS*, pages 1007–1014, 2008.

[6] G. Greco, E. Malizia, L. Palopoli, and F. Scarcello. On the complexity of core, kernel, and bargaining set. *Artificail Intelligence*, 175(12-13):1877–1910, 2011.

[7] S. Ieong and Y. Shoham. Marginal contribution nets: a compact representation scheme for coalitional games. In *ACM EC*, pages 193–202, 2005.

[8] T. Michalak, D. Marciniak, M. Szamotulski, T. Rahwan, M. Wooldridge, P. McBurney, and N. R.Jennings. A logic-based representation for coalitional games with externalities. In *AAMAS*, pages 125–132, 2010.

[9] T. Michalak, J. Sroka, T. Rahwan, M. Wooldridge, P. McBurney, and N. R. Jennings. A distributed algorithm for anytime coalition structure generation. In *AAMAS*, pages 1007–1014, 2010.

[10] N. Ohta, V. Conitzer, R. Ichimura, Y. Sakurai, A. Iwasaki, and M. Yokoo. Coalition structure generation utilizing compact characteristic function representations. In *CP*, pages 623–638, 2009.

[11] T. Rahwan, T. Michalak, E. Elkind, P. Faliszewski, J. Sroka, M. Wooldridge, and N. R. Jennings. Constrained coalition formation. In *AAAI*, 2011.

[12] T. Rahwan, T. Michalak, N. R. Jennings, M. Wooldridge, and P. McBurney. Coalition structure generation in multi-agent systems with positive and negative externalities. In *IJCAI*, pages 257–263, 2009.

[13] T. Rahwan, S. D. Ramchurn, N. R. Jennings, and A. Giovannucci. An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research (JAIR)*, 34:521–567, 2009.

[14] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1-2):1–54, 2002.

[15] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1-2):209–238, 1999.

[16] S. Ueda, M. Kitaki, A. Iwasaki, and M. Yokoo. Concise characteristic function representations in coalitional games based on agent types. In *IJCAI*, pages 393–399, 2011.

[17] D. Y. Yeh. A dynamic programming approach to the complete set partitioning problem. *BIT Numerical Mathematics*, 26(4):467–474, 1986.

Session 2E
Game Theory II

Short Sight in Extensive Games

Davide Grossi
University of Liverpool
Department of Computer Science
Ashton Street, Liverpool, L69 3BX, UK
d.grossi@liverpool.ac.uk

Paolo Turrini
University of Luxembourg
Computer Science and Communication
6, rue Richard Coudenhove - Kalergi,
Luxembourg L-1359, Luxembourg
paolo.turrini@uni.lu

ABSTRACT

The paper introduces a class of games in extensive form where players take strategic decisions while not having access to the terminal histories of the game, hence being unable to solve it by standard backward induction. This class of games is studied along two directions: first, by providing an appropriate refinement of the subgame perfect equilibrium concept, a corresponding extension of the backward induction algorithm and an equilibrium existence theorem; second, by showing that these games are a well-behaved subclass of a class of games with possibly unaware players recently studied in the literature.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Theory

Keywords

Extensive games, bounded rationality

1. INTRODUCTION

In the past decade the multi agent systems (MAS) community has witnessed several attempts to relax the strong assumptions underpinning game-theoretical models, such as common knowledge of the game structure, logical omniscience and unbounded computational power, to mention a few. Along these lines Joseph Halpern's invited talk at AAMAS 2011—*Beyond Nash-Equilibrium: Solution Concepts for the 21st Century*—highlighted several research challenges that arise when attempting to provide more realistic versions of the Nash equilibrium solution concept. Among those challenges, the issue of unawareness seems to stand out, viz. the observation that in real games, like for instance chess, players take decisions even if they cannot possibly have access to the whole game form. Halpern himself extensively contributed to the research on players' unawareness: in [4] and its extension [5], a game-theoretical analysis of unawareness in extensive games is presented, where players have access to only part of the terminal histories of a game tree as they ignore, at some nodes, some of the actions available to their fellow players. The same phenomenon has been studied, although by different means, by Yossi Feinberg in [1, 2].

All the aforementioned models of unawareness in games make a common assumption: players might be unaware of some branches of the game tree, but they do have access to a subset of the terminal histories, that is, they have a full representation of at least some possible endings of the game. With the present work we would like to push Halpern's stance further, by lifting this assumption and present a model of players who not only might not see a part of the terminal nodes of a game tree but who might not even see any such nodes. As happens in real games like chess, but also in a number of occasions where individuals are confronted with a large game structure, decisions are taken on the basis of a stepwise evaluation of foreseeable intermediate positions. As the game proceeds, it often reveals earlier decisions to be wrong.[1] The following example provides a concrete motivating scenario representing this special kind of unawareness, which we will be calling *short sight.*

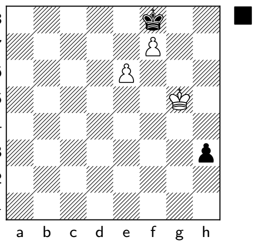


Figure 1: Black to move

Example 1 (A chess scenario) *In Figure 1 Black is to move. He has three options at his disposal: moving the black king to g7 (shortly ♚g7), moving it to e7 (♚e7), or moving the pawn one square further to h2 (h2). Let us assume that Black has to move under pressing time constraints or that he is not well-versed in evaluating key positions on the chessboard. He will then take into consideration only a few possible developments of the play—for instance what he would be able to reach in two moves (i.e., some plays up to two steps*

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

[1]To say it with [9], "Chess is a draw that is only made competitive by human error" .

ahead)—and he will base his decisions on somewhat 'coarse' evaluations—for instance, gaining material advantage.

If this is the case, in a situation such as the one displayed in Figure 1, he will prefer to queen his pawn as quickly as possible.[2] *Comparing the moves ♔g7, ♔e7 and h2, the latter clearly leads to material advantage while the formers do not. So Black will go for h2. However after h2 White can move its king to f6 (♔f6). Now Black is in trouble because after the white king is in f6 Black has only one move at its disposal — he must queen his pawn (h1) — as ♔e7 and ♔g7 are now illegal. Black's material advantage in the resulting position (one queen against two pawns) is no consolation: after e7 Black is checkmated.*

In the example Black loses for two reasons: 1) he has partial view even on the immediate development of the game; 2) he bases his decision on an evaluation criterion—reaching material advantage—which turns out to be counter-productive. These observations exemplify the characteristics of short sight in extensive games: 1) players may be aware of only part of the game structure and may not be able to calculate the consequences of their actions up to the terminal nodes; 2) at each choice point, players base their decisions evaluating the positions they can foresee according to (possibly faulty) criteria. The paper will incorporate the characteristic features of short sight in a standard treatment of extensive games, studying their properties and their relation with models of players' unawareness to be found in the literature—in particular the ones in [4].

Outline of the paper. Section 2 introduces the basic terminology and facts to be used later on in the paper. It mainly concernes the notion of extensive game and preference relation and it presents standard solution concepts, such as the subgame perfect equilibrium. Section 3 equips extensive games with a description of players' limited view at each history and presents corresponding solution concepts for the new models. In particular it defines a backward induction algorithm for games with short sight and proves an equilibrium existence theorem for this class of games. Section 4 discusses the relation between games with short sight and games with awareness as studied by Halpern and Rêgo. Concretely, it shows that games with short sight are a special type of games with awareness. Section 5 concludes the paper pointing to several possible developments.

2. PRELIMINARIES

The section introduces the basic terminology and notation to be used in the rest of the paper.

2.1 Game forms and games

The structures we will be working with are extensive games which, unlike the games in strategic or normal form, take the sequential structure of decisions into account [7]. We start out introducing extensive games forms of perfect information (henceforth simply "extensive game forms" or "game forms"), where players have full knowledge of the possible courses of events. The following definition is adapted from [7].

[2]The black pawn reaching h1 can be queened, i.e. turned into a strong major piece, giving its owner an often decisive advantage.

Definition 1 (Extensive game forms) *An* extensive game form *is a tuple $\mathcal{G} = (N, H, \mathrm{t}, \Sigma_i, o)$ where:*

- *N is a non-empty set of players;*
- *H is a non-empty set of sequences, called histories, such that: 1) The empty sequence $\emptyset$ is a member of H; 2) If $(a^k)_{k=1,\ldots,K} \in H$ and $L < K$ then $(a^k)_{k=1,\ldots,L} \in H$; 3) If an infinite sequence $(a^k)_{k=1}^{\omega}$ is s.t. $(a^k)_{k=1,\ldots,L} \in H$ for every $L < \omega = |\mathbb{N}|$ then $(a^k)_{k=1}^{\omega} \in H$.*

  A history $h \in H$ is called terminal *if it is infinite or it is of the form $(a^k)_{k=1,\ldots,K}$ with $K < \omega$ and there is no a^{K+1} such that $(a^k)_{k=1,\ldots,K+1} \in H$. The set of terminal histories is denoted Z. Each component of a history is called an* action. *The set of all actions is denoted A. The set of actions following a history h is denoted with $A(h)$. Formally $A(h) = \{a \mid (h, a) \in H\}$. If h is a prefix of h' we write $h \lhd h'$.*
- *$\mathrm{t} : H \backslash Z \to N$ is a function, called* turn function*, assigning players to non-terminal histories, with the idea that player i moves at history h whenever $\mathrm{t}(h) = i$;*
- *Σ_i is a non-empty set of strategies $\sigma_i : \{h \in H \backslash Z \mid \mathrm{t}(h) = i\} \to A$ for each player i that assign an action to any non-terminal history whose turn to play is i's; we refer to $\sigma_{\mathrm{t}(h)}(h)$ as the action* prescribed *by strategy σ at history h for the player who moves at h;*
- *$o : \prod_{i \in N} \Sigma_i \to Z$ is a bijective* outcome *function from strategy profiles to terminal histories.*

For any set of histories $A \subseteq H$ we denote $l(A)$ the length of its longest history. The notation can also be used with game forms, where $l(\mathcal{G}) = l(H)$, for H being the set of histories of game form $\mathcal{G}$. If H is a finite set $\mathcal{G}$ is called a *finite* game form. Extensive game forms equipped with preference relations, i.e. a family of orders on terminal histories for each player, are referred to as extensive games (or simply as games).

Definition 2 (Extensive games) *An extensive game is a tuple $\mathcal{E} = (\mathcal{G}, \succeq_i)$ where $\mathcal{G}$ is an extensive game form and $\succeq_i \subseteq Z^2$ is a total preorder*[3] *over Z, for each player i.*

An extensive game $\mathcal{E} = (\mathcal{G}, \succeq_i)$ is called *finite* if $\mathcal{G}$ is finite.

2.2 Preferences and evaluation criteria

In Definition 2 players' preferences are given by a total preorder over the set of terminal nodes. However situations such as the one described in Example 1 suggest that, in presence of short sight, decisions need to be taken even when terminal nodes are not accessible. For this reason we assume here that players hold preferences about foreseeable intermediate nodes according to general criteria which remain stable throughout the game. The idea is that players are endowed with some kind of 'theory' that allows them to conceptualize and evaluate game positions. For instance, in Example 1 *Black* evaluates the positions that he can calculate according to the general criterion of material advantage.

[3]I.e., a reflexive, transitive and total binary relation.

2.2.1 *Priority sequences*

To model the intuition above we follow a simple strategy. We take evaluation criteria to consist of preferences defined over properties of game positions, and we take properties to be sets of game positions, i.e., sets of histories.

Definition 3 (Priority sequences) *Let $\mathcal{G} = (N, H, \mathrm{t}, \Sigma_i, o)$ be an extensive game form. A priority sequence, or P-sequence, for $\mathcal{G}$ is a tuple $P = (\mathcal{H}, \succ)$ where:*

- *$\mathcal{H} \subseteq \wp(H)$ and $\mathcal{H}$ is finite, i.e., the set of properties $\mathcal{H}$ is a finite set of sets of histories. Elements of $\mathcal{H}$ are denoted $\mathbf{H}, \mathbf{H}', \ldots$.*
- *$\succ \subseteq \mathcal{H}^2$ is a strict linear order[4] on the properties in $\mathcal{H}$. To say that $\mathbf{H}$ is preferred to $\mathbf{H}'$, for $\mathbf{H}, \mathbf{H}' \in \mathcal{H}$, we write: $\mathbf{H} \succ \mathbf{H}'$.*

P-sequences express a fixed priority between a finite set of relevant criteria. In our understanding they represent a general theory that a player can use to assess game positions. P-sequences and their generalisation to graphs have been object of quite some recent studies in the logic of preference, such as [6] from which Definition 3 is adapted. Given a P-sequence, a preference over histories can be derived in a natural way:

Definition 4 (Preferences) *Let $\mathcal{G} = (N, H, \mathrm{t}, \Sigma_i, o)$ be an extensive game form and $P = (\mathcal{H}, \succ)$ a P-sequence for $\mathcal{G}$. The preference relation $\succeq^P \subseteq H^2$ over the set of histories of $\mathcal{G}$ induced by P is defined as follows:*

$$h \succeq^P h' \iff \forall \mathbf{H} \in \mathcal{H} : [\text{ IF } h' \in \mathbf{H} \text{ THEN } h \in \mathbf{H} \text{ OR } \exists \mathbf{H}' \in \mathcal{H} : [h \in \mathbf{H}' \text{ AND } h' \notin \mathbf{H}' \text{ AND } \mathbf{H}' \succ \mathbf{H}]]$$

In words, a history h is at least as good as a history h' according to P, if and only if, either all properties occurring in P that are satisfied by h' are also satisfied by h or, if that is not the case and there is some property that h' has but h has not, then there exists some other better property which h satisfies and h' does not. This 'recipe' yields preferences of a standard type:

Fact 1 *Let $\mathcal{G}$ be an extensive game form and $P = (\mathcal{H}, \succ)$ a P-sequence for $\mathcal{G}$. The relation $\succeq^P$ has the following properties: 1) It is a total pre-order; 2) $\succeq^P$ contains at most $2^{|\mathcal{H}|}$ sets of equally preferred elements.*[5]

PROOF (SKETCH). 1. That $\succeq^P$ is reflexive follows directly from Definition 4. Transitivity is established by the following argument: assume $h \succeq^P h'$ and $h' \succeq^P h''$. By Definition 4 we have four possible cases: i) all properties satisfied by h' are also satisfied by h and all properties satisfied by h'' are also satisfied by h', hence $h \succeq^P h''$; ii) all properties satisfied by h' are also satisfied by h and for some property $\mathbf{H}$ enjoyed by h'' but not by h' there exists another property $\mathbf{H}'$ such that $\mathbf{H}' \succ \mathbf{H}$ and h' satisfies $\mathbf{H}$ but h'' does not. Hence for some property $\mathbf{H}$ enjoyed by h but not by h'' there exists another property $\mathbf{H}'$ such that $\mathbf{H} \succ \mathbf{H}'$ and h satisfies $\mathbf{H}$ but h'' does not, from which we conclude $h \succeq^P h''$. iii) More schematically, for all $\mathbf{H}$: $\exists \mathbf{H}' \in \mathcal{H} : [h' \in \mathbf{H}' \text{ AND } h'' \notin \mathbf{H} \text{ AND } \mathbf{H}' \succ \mathbf{H}]]$ and $\forall \mathbf{H} \in \mathcal{H} : [\text{ IF } h' \in \mathbf{H} \text{ THEN } h \in \mathbf{H}]$. The proof is analogous to the one of ii). iv) For all $\mathbf{H}$: $\exists \mathbf{H}' \in \mathcal{H} : [h \in \mathbf{H}' \text{ AND } h' \notin \mathbf{H} \text{ AND } \mathbf{H}' \succ \mathbf{H}]]$ and $\exists \mathbf{H}'' \in \mathcal{H} : [h' \in \mathbf{H}'' \text{ AND } h'' \notin \mathbf{H}'' \text{ AND } \mathbf{H}'' \succ \mathbf{H}]]$ follows from the transitivity of relation $\succ$ (Definition 3). As for totality, suppose not $h \succeq^P h'$. But then, by totality of $\succ$ (Definition 3) $\exists \mathbf{H} \in \mathcal{H} : [h' \in \mathbf{H} \text{ AND } h \notin \mathbf{H} \text{ AND } \forall \mathbf{H}' \in \mathcal{H} : [h \in \mathbf{H}' \text{ AND } h' \notin \mathbf{H}' \text{ IMPLIES } \mathbf{H} \succ \mathbf{H}']]$, which implies that $h' \succeq^P h$.

2) Equivalence classes in $\succeq^P$ are determined by the set of properties in $\mathcal{H}$ that they satisfy, hence by elements of $\wp(\mathcal{H})$. As some of these sets might be empty, $2^{|\mathcal{H}|}$ is an upper bound. $\square$

[4]I.e. an irreflexive, transitive, asymmetric and total binary relation.

[5]I.e., sets of elements h, h' such that $h \succeq^P h'$ and $h' \succeq^P h$.

Intuitively, P-sequences yield total preorders consisting of a finite set of equally preferred elements which form a linear hierarchy from the set of most preferred elements to the set of least preferred elements.

Example 2 *As an illustration, recall Example 1. We could model Black's evaluation criteria by the following simple P-sequence (let* `cm` *denote the set of histories where White is checkmated,* `dr` *the set of histories where the game is a draw, and* `ma` *the set of histories where Black has material advantage):* $\mathtt{cm} \succ \mathtt{dr} \succ \mathtt{ma}$ *This P-sequence yields the following total preorder over histories:*[6]

$$\begin{tikzcd}
\mathtt{cm} \cap -\mathtt{dr} \cap \mathtt{ma} \arrow[d, dash] \\
\mathtt{cm} \cap -\mathtt{dr} \cap -\mathtt{ma} \arrow[d, dash] \\
-\mathtt{cm} \cap \mathtt{dr} \cap \mathtt{ma} \arrow[d, dash] \\
-\mathtt{cm} \cap \mathtt{dr} \cap -\mathtt{ma} \arrow[d, dash] \\
-\mathtt{cm} \cap -\mathtt{dr} \cap \mathtt{ma} \arrow[d, dash] \\
-\mathtt{cm} \cap -\mathtt{dr} \cap -\mathtt{ma}
\end{tikzcd}$$

where we have assumed that no history can be a checkmate and a draw at the same time. In words, Black prefers most of all positions where White is checkmated and at the same time he retains material advantage, then positions where White is checkmated without material advantage, and so according to the above P-sequence. The worst positions are the ones where none of the properties occurring in the P-sequence are satisfied.

It is worth observing that the elements of a P-sequence can be represented by set-theoretic compounds of properties [7].

[6]The total preorder is represented as a Hasse diagram consisting of linearly ordered equivalence classes. Standard set-theoretic notation for inclusion and complementation is used.

[7]As an anonymous reviewer pointed out, there may be situations in which two properties $\mathbf{H}$ and $\mathbf{H}'$ that, when occurring together, outweigh a third one $\mathbf{H}''$, while $\mathbf{H}''$ would be preferred over both $\mathbf{H}$ and $\mathbf{H}'$ when they occur alone (e.g., centre control *together with* an exposed opponent's king may outweigh material disadvantage). In our framework this is handled by stating that $\mathbf{H} \cap \mathbf{H}' \succ \mathbf{H}'' \succ \mathbf{H} \cup \mathbf{H}'$.

The link to logic should here be evident as sets of histories—our properties—could be seen as denotations of formulae in some logical language (e.g. propositional logic). Our exposition abstracts from the logical aspect which could, however, add a further interesting syntactic dimension to our account.

2.2.2 *Games with priorities*

Henceforth we will be working with game forms that are endowed with a family of P-sequences, one for each player:

Definition 5 (Prioritized games) *Let $\mathcal{G}$ be a game form and let P_i be a family of P-sequences for $\mathcal{G}$, one for each player $i \in N$. A* prioritized game *is a tuple $\mathcal{G}^P = (\mathcal{G}, P_i)$.*

Clearly, each prioritized game $\mathcal{G}^P = (\mathcal{G}, P_i)$ defines a game in extensive form (Definition 2) $\mathcal{E}_{\mathcal{G}^P} = (\mathcal{G}, Z^2 \cap \succeq^{P_i})$. So, when attention is restricted to terminal histories, prioritized games yield standard extensive form games. What they add to the them is information by means of which players can systematically rank non-terminal histories also without having access to terminal histories.

2.3 Subgame-perfect equilibrium

In this section we adapt the notion of subgame-perfect equilibrium to prioritized games. The adaptation is straightforward since each prioritized game univocally determines an extensive one. It is nevertheless worth it to introduce all the notions in details, as they will be our stepping stone for the definition of an analogous solution concept in games with short sight. We first introduce the notion of subgame.

Definition 6 (Subgames of prioritized games) *Take a prioratized game $\mathcal{G}^P = ((N, H, \mathrm{t}, \Sigma_i, o), P_i)$. Its subgame from h is a prioritized game $\mathcal{G}^P_h = ((N|_h, H|_h, \mathrm{t}|_h, \Sigma_i|_h, o|_h), P_i|_h)$ such that: 1) $H|_h$ is the set of sequences h' for which $(h, h') \in H$; 2) $\Sigma_i|_h$ is the set of strategies for each player available at h. It consists of elements $\sigma_i|_h$ such that $\sigma_i|_h(h') = \sigma_i(h, h')$ for each $h' \in H|_h$ with $\mathrm{t}(h, h') = i$; 3) $\mathrm{t}|_h$ is such that $\mathrm{t}|_h(h') = \mathrm{t}(h, h')$ for each $h' \in H|_h$; 4) $o|_h : \prod_{i \in N} \Sigma_i|_h \to Z|_h$ is the outcome function of $\mathcal{G}^P_h$, where $Z|_h$ is the set of sequences h' for which $(h, h') \in Z$; 5) $P_i|_h = P_i$.*

Now we are ready to introduce subgame perfect equilibria.

Definition 7 (Subgame perfect equilibrium) *Let $\mathcal{G}^P$ be a finite prioritized game. A strategy profile σ^* is a subgame perfect equilibrium if for every player $i \in N$ and every non-terminal history $h \in H \backslash Z$ for which $\mathrm{t}(h) = i$ we have that:*

$$o|_h(\sigma^*_i|_h, \sigma^*_{-i}|_h) \succeq^{P_i} o|_h(\sigma_i, \sigma^*_{-i}|_h)$$

for every strategy σ_i available to player i in the subgame $\mathcal{G}^P_h$ that differs from $\sigma^*_i|_h$ only in the action it prescribes after the initial history of $\mathcal{G}^P_h$.

The definition of subgame perfect equilibrium is normally given in its stronger version, without the requirement that σ_i for player i in the subgame $\mathcal{G}^P_h$ differs from $\sigma^*_i|_h$ only in the action it prescribes after the initial history of $\mathcal{G}^P_h$. However the formulation we have given is equivalent to the stronger version for the case of finite games, as proved in [7, Lemma 98.2]. This property of the subgame perfect equilibria is known as *the one deviation property.*

By Kuhn's theorem[8] we can then conclude that all finite prioritized games have a subgame perfect equilibrium.

Remark 1 *The existence of subgame perfect equilibria in finite extensive games is usually proven constructively via the well-known backward induction (BI) algorithm. It might be worth recalling that the algorithm solves the game by extending the total preorder on the terminal histories of the game to a total preorder over all histories, where for every player each history is as preferred as the terminal history it leads to under the assumption that the other players play 'rationally'. So the result of the algorithm is a total preorder over all histories consisting of a finite set of equivalence classes, viz. the sort of preference structures also determined by P-sequences (Fact 1). The key difference, however, is that while the order determined by BI is consistent with the order on the terminal nodes, in the sense that keeping on choosing the best option guarantees the best outcome in the game, no such guarantee exist in the order yielded by a P-sequence—as Example 1 neatly shows.*

3. SHORT SIGHT IN GAMES

In this section we introduce and discuss the notion that has motivated the present work: short sight.

3.1 Players' sights

The following definition introduces a simple device to capture what and how deep each player can see in the game at each choice point.

Definition 8 (Sight function) *Let $\mathcal{G}^P = ((N, H, \mathrm{t}, \Sigma_i, o), P_i)$ be a prioritized game. A (short) sight function for $\mathcal{G}^P$ is a function $\mathrm{s} : H \backslash Z \to 2^H \backslash \emptyset$ associating to each non-terminal history h a finite subset of all the available histories at h. That is: 1) $\mathrm{s}(h) \in 2^{H|_h} \backslash \emptyset$ and $|\mathrm{s}(h)| < \omega$, i.e. the sight at h consists of a finite nonempty set of histories extending h; 2) $h' \in \mathrm{s}(h)$ implies that $h'' \in \mathrm{s}(h)$ for every $h'' \lhd h'$, i.e., sight is closed under prefixes.*

Intuitively, the function associates to any choice point those histories that the player playing at that choice point can see. Notice that how this set of histories is determined is left open. In other words, the set constitutes the view that the player playing at that non-terminal history has of the remaining of the game. It could be, for instance, all the histories of length at least d, or all histories that start with a given action a, or similar constraints.

The intuition is that $\mathrm{s}(h)$ is the limited view of $\mathrm{t}(h)$ after history h. Such intuition is supported by the fact that $\mathrm{s}(h)$ inherits the moves and the turns from $\mathcal{G}^P$ but not necessarily the terminal nodes. That the view is limited can be noticed by the conditions required in Definition 8, which together imply that $l(\mathrm{s}(h)) < \omega$, i.e. players can only see finitely many steps ahead. Several extra conditions, besides the one given in Definition 8, might be natural for short sight, e.g.: requiring that the sight increases as the play proceeds, in the sense that what player i can see from h is at least as much as from any history hh'. The present work will not deal with these extra conditions and will limit itself to a general account.

We now define the class of games with short sight.

[8]We adopt the terminology of [7, Proposition 99.2] and refer to the result stating that every finite extensive game has a subgame perfect equilibrium as *Kuhn's theorem.*

Definition 9 (Games with short sight) *A game with short sight is a tuple $\mathcal{S} = (\mathcal{G}^P, \mathrm{s})$ where $\mathcal{G}^P$ is a prioritized game and* s *a sight function for $\mathcal{G}^P$.*

It is clear that each game with short sight yields a family of finite extensive games, one for each non-terminal history:

Fact 2 *Let $\mathcal{S} = (\mathcal{G}^P, \mathrm{s})$ be a prioritized game with short sight, with $\mathcal{G}^P = ((N, H, \mathrm{t}, \Sigma_i, o), P_i)$. Let also h be a finite non-terminal history. Consider the tuple:*

$$\mathcal{E}\lceil_h = (N\lceil_h, H\lceil_h, \mathrm{t}\lceil_h, \Sigma_i\lceil_h, o\lceil_h, \succeq_i \lceil_h)$$

where: 1) $N\lceil_h = N$; 2) $H\lceil_h = \mathrm{s}(h)$. The set $Z\lceil_h$ denotes the histories in $H\lceil_h$ of maximal length, i.e., the terminal histories in $H\lceil_h$; 3) $\mathrm{t}\lceil_h = H\lceil_h \backslash Z\lceil_h \to N$ so that $\mathrm{t}\lceil_h(h') = \mathrm{t}(h, h')$; 4) $\Sigma_i\lceil_h$ is the set of strategies for each player available at h and restricted to $\mathrm{s}(h)$. It consists of elements $\sigma_i\lceil_h$ such that $\sigma_i\lceil_h(h') = \sigma_i(h, h')$ for each $(h', \sigma_i(h, h')) \in H\lceil_h$ with $\mathrm{t}\lceil_h(h') = i$; 5) $o\lceil_h$: $\prod_{i\in N} \Sigma_i\lceil_h \to Z\lceil_h$; 6) $\succeq_i \lceil_h = \succeq^{P_i} \cap (Z\lceil_h)^2$. Tuple $\mathcal{E}\lceil_h$ is a finite extensive game.

Remark 2 *It is worth noticing that each* finite *extensive game $\mathcal{E}_{\mathcal{G}^P}$ determined by a prioritized game $\mathcal{G}^P$ (recall Definition 5) is equivalent (modulo the sight function) to the game with short sight built on $\mathcal{G}^P$ such that, for each h, $\mathcal{E}_{\mathcal{G}^P}\lceil_h = \mathcal{E}_{\mathcal{G}^P}|_h$. That is, at each non-terminal history, the game determined by the sight function corresponds to the whole subgame at h.*

3.2 Solving games with short sight

In games with short sight the course of the play is such that at each node players are confronted with decisions to be taken on the grounds of what they can foresee of the game. The purpose of this section is to provide a model of rationality for such situations, i.e. what players should do given the history of the play and their sight.

3.2.1 Subgame perfect equilibria

As we are dealing with self-interested agents, it is natural to think that they will try to get the most out of the information they possess, choosing their best strategy at each choice node. This leads us to a simple adaptation of the notion of subgame perfect equilibrium (Definition 7).

Definition 10 (Sight-compatible subgame perfection) *Take a game with short sight $\mathcal{S} = (\mathcal{G}^P, \mathrm{s})$ and, for each finite history h, let $\mathcal{E}\lceil_h$ be the extensive game yielded by* s *at h (as defined in Fact 2). A sight-compatible subgame perfect equilibrium of $\mathcal{S}$ is a profile of strategies $\sigma^* \in \prod_{i\in N} \Sigma_i$ such that for every nonterminal history h there exists a strategy profile $\sigma\lceil_h$ that is a subgame perfect equilibrium of $\mathcal{E}\lceil_h$ and such that $\sigma_{\mathrm{t}(h)}\lceil_h(h) = \sigma^*_{\mathrm{t}(h)}(h)$.*

Three aspects of the equilibrium definition are worth mentioning. First, each restriction $\mathcal{E}\lceil_h$ prunes the game tree at the bottom (considering the extensions of h) and at the top (considering only the sight-compatible extensions of h). Second, each player i determines his best move supposing that his opponents behave rationally with respect to their P-sequences and relative to the part of the game that i can see. This might be considered a conservative—or safe, depending on the circumstances—way for i to play, by attributing to the opponents the ability to see at least as much as i sees. Third, the definition of subgame perfect equilibrium in games with short sight does not require an explicit finiteness assumption. A finiteness assumption—the finiteness of the histories constituting the sight—is built in Definition 8. This brings us to the next section.

3.2.2 An equilibrium existence theorem

Let us start with the following observation:

Fact 3 *Let $\mathcal{S} = (\mathcal{G}^P, \mathrm{s})$ be a game with short sight and h one of its finite non-terminal histories. Then $\mathcal{E}\lceil_h$ has a subgame perfect equilibrium.*

PROOF. The fact is a direct consequence of Fact 2 and Kuhn's theorem [7, Proposition 99.2]. $\square$

We can now prove the existence of sight-compatible subgame perfect equilibria (Definition 10).

Theorem 3 (Equilibrium existence) *Every game with short sight has a sight-compatible subgame perfect equilibrium.*

PROOF. Let $\mathcal{S} = (\mathcal{G}^P, \mathrm{s})$ be a game with short sight and let σ^* be a strategy profile such that, for each non-terminal history h: $\sigma^*_{\mathrm{t}(h)}(h) = \sigma^{BI(\mathcal{E}\lceil_h)}_{\mathrm{t}(h)}$ where $\sigma^{BI(\mathcal{E}\lceil_h)}$ denotes the strategy profile constructed by the standard backward induction algorithm on the extensive game $\mathcal{E}\lceil_h$ determined by the sight function at history h. The result follows then directly by the construction—via backward induction—of subgame perfect equilibria for each $\mathcal{E}\lceil_h$ (Kuhn's theorem) as $\sigma^{BI(\mathcal{E}\lceil_h)}_{\mathrm{t}(h)}$ is the action dictated to player $\mathrm{t}(h)$ by its backward induction strategy and therefore the action dictated by a subgame perfect equilibrium of $\mathcal{E}\lceil_h$. $\square$

3.2.3 An algorithm for solving games with short sight

By building on the standard backward induction algorithm (BI), we can define an algorithm which solves each finite game with short sight by constructing a terminal history, the one determined a sight-compatible subgame perfect equilibrium of the game.

Definition 11 (BI-path in games with short sight)

Input: *A finite game with short sight $\mathcal{S} = (\mathcal{G}^P, \mathrm{s})$*

Output: *A terminal history $(x_0, \ldots, x_n)$ of $\mathcal{G}^P$*

```
Method: 1. Define $h := \emptyset$;
        2. Run $BI$ over $\mathcal{E}\lceil_h$ and set $h := \left(h, \sigma^{BI(\mathcal{E}\lceil_h)}_{\mathrm{t}(h)}\right)$;
        3. If $h \in Z$ then return $h$, otherwise repeat step 2.
```

It is easy to see that the algorithm terminates and constructs indeed a history consisting of actions dictated by a sight-compatible subgame perfect equilibrium. Intuitively, the algorithm starts at the root and solves $\mathcal{E}\lceil_\emptyset$. This yields a terminal history in $Z\lceil_\emptyset$, and their initial fragments of length 1 are taken as the first moves of the histories returned by the algorithm. Each of these first moves determine, in turn, as many extensive games via the sight function. These are solved in the same way, determining a set of histories of length 2, and so on, until terminal histories of $\mathcal{G}^P$ are built.

4. SHORT SIGHT AND UNAWARENESS

This section is devoted to establishing the precise relationship between games with short sight and games with possibly unaware players elaborated by Halpern and Rêgo in [4]. As already pointed out, the models focused upon in [4] feature players that can always observe at least some of the terminal histories of the actual game being played. In the same paper, in order to overcome this limitation, Halpern and Rêgo generalize their models to allow players to hold false beliefs about the game being played although, it must be mentioned, they do not provide an equilibrium analysis of that class of games. Essentially, at each node of a game each player might believe to be playing a completely different game from the one that he or she is actually playing. These generalized models are extremely abstract and can incorporate several forms of unawareness. Even though the intuitive understanding of short sight is rather different from that of false belief, the models in [4] can be formally related to our models. To establish this relationship we proceed as follows.

First, we formally introduce games with possibly unaware players and lack of common knowledge of the underlying game, the most general model of unawareness provided in [4]. We will refer to this class of models simply as *games with awareness* (Subsection 4.1). Second, we provide a canonical representation of games with short sight as games with awareness. In short, we are going to build a class of the latter models where, at each position of the actual game being played, players *believe to be playing a game that corresponds to their own sight.* We show, moreover, that the canonical representation is of the right kind, i.e. it obeys the axioms of the general models of Halpern and Rêgo (Subsection 4.2). Third, we provide the axioms that characterize games with short sight as games with awareness (Subsection 4.3).

4.1 Games with awareness

Halpern and Rêgo work with finite extensive games endowed with information sets and probability measures [4]. As the games structures dealt with in our paper do not model epistemic aspects such as knowledge and belief, the comparison to which this section is devoted will concern the somewhat more fundamental level of the finite extensive games with perfect information upon which Halpern and Rêgo base their models.

To each extensive game $\mathcal{E} = ((N, H, \mathrm{t}, \Sigma_i, o), \succeq_i)$, [4] associates an *augmented game* ${}^{+}\mathcal{E}$ that specifies the level of awareness of each player at each node of the original game. The following definition is adapted from [4].

Definition 12 (Augmented game) *Let $\mathcal{E} = (\mathcal{G}, \succeq_i)$ be a finite extensive game and, for each history h (not necessarily belonging to the set of histories of $\mathcal{G}$), let $\overline{h}$ be the subsequence of h consisting of the moves in h that are made by actions available in $\mathcal{G}$. The* augmented game ${}^{+}\mathcal{E} = (((N, H, \mathrm{t}, \Sigma_i, o), \succeq_i), Aw_i)$ *based on $\mathcal{G}$ is such that:*

A1 $(N, H, \mathrm{t}, \Sigma_i, o), \succeq_i)$ *is a finite extensive game;*

A2 $Aw_i : H \to 2^{H'}$ *is the awareness function of each player i, that maps each history to a set of histories (in $2^{H'}$) of some arbitrary finite extensive game $\mathcal{E}'$. For each $h \in H$ the set $Aw_i(h)$ consists of histories in H' and their prefixes.*

A11 $\{\overline{z} \mid z \in Z\} \subseteq Z$, *i.e. the terminal histories of the game ${}^{+}\mathcal{E}$ correspond to terminal histories of $\mathcal{E}$; moreover if z' is a terminal history of ${}^{+}\mathcal{E}$ then $z' \in Z$, i.e. terminal histories of which players are aware are terminal histories of the game $\mathcal{E}$ upon which ${}^{+}\mathcal{E}$ is based.*

A12 *for each terminal history $z \in Z$ such that $\overline{z} \in Z$ we have that $z \succeq_i \overline{z}$ and $\overline{z} \succeq_i z$ for each $i \in N$, i.e. players' preferences come from game $\mathcal{E}$ upon which ${}^{+}\mathcal{E}$ is based.*

The items in the definition keep the original names of axioms A1, A2, A11 and A12 given in [4] for games with lack of common knowledge.

We can now formally introduce a game with awareness in its most general form.

Definition 13 (Games with awareness) *Let $\mathcal{E}$ be a finite extensive game. A* game with awareness *based on $\mathcal{E}$ is a tuple $\mathcal{E}^{Aw} = (\Gamma, \mathcal{E}^m, \mathcal{F})$, where:*

- *Γ is a countable set of augmented games each one based on some (possibly different) game $\mathcal{E}'$;*
- *$\mathcal{E}^m$ is a distinguished augmented game based on $\mathcal{E}$;*
- *$\mathcal{F}$ is a mapping that associates to each augmented game ${}^{+}\mathcal{E}' \in \Gamma$ and history h' of ${}^{+}\mathcal{E}'$ an augmented game $\mathcal{E}_{h'}$. This game is the game the player whose turn is to play believes to be the true game when the history is h'.*[9]

The definition spells out the crucial feature of a game with awareness, namely the fact that each player at each history is associated to a game that he belives to be the current game. This can be distinct from the current game being played, which is instead observed by an omniscent modeller. Specifically, while each ${}^{+}\mathcal{E}'$ is the point of view of some player at some history (the precise relation is given by the $\mathcal{F}$ mapping), $\mathcal{E}^m$ is the point of view of the omniscent modeller, who can actually see the game that is being played and the players' awareness level. Definition 13 is extremely abstract and can be refined by imposing several reasonable constraints, especially with respect to $\mathcal{E}^m$, the point of view of the modeller. The following definition, adapted from [4], takes care of that.

Definition 14 (Games with awareness: constraints) *The class of games with awareness is refined by the following constraints, for each $\mathcal{E}^{Aw} = (\Gamma, \mathcal{E}^m, \mathcal{F})$:*

M1 $N^m = N$, *i.e. the modeller is aware of all the players;*

M2 $A \subseteq A^m$ *and* $\{\overline{z} : z \in Z^m\} = Z$, *i.e. the modeller is aware of all the moves available to the players and knows the terminal histories of the game;*

M3 *If* $\mathrm{t}^m(h) \in N$ *then* $A^m(h) = A(\overline{h})$, *i.e. the modeller is aware of the possible courses of the events;*

C1 $\{\overline{h'} \mid h' \in H_h\} = Aw_i(h)$, *i.e. the awareness function shows exactly the histories that can be observed.*

The constraints just discussed hold for all games with awareness. The following part lays a first bridge between these structures and games with short sight.

4.2 Canonical representation

In [8] a canonical representation is provided of a finite extensive game as a game with awareness. For the present purposes, which are not concerned with epistemic aspects, a finite extensive game $\mathcal{E}$ is representable as a tuple $(\{\mathcal{E}^m\}, \mathcal{E}^m, \mathcal{F})$

[9]Henceforth, to reduce clutter in notation, we use the subscript $_{h'}$ to index the elements of game tuple $\mathcal{E}_{h'}$, i.e. the game that player $\mathrm{t}(h')$ believes to be playing at history h'. For instance $H_{h'}$ is the set of histories that player $\mathrm{t}(h')$ believes to be the set of histories that are available when he is in h'.

where $\mathcal{E}^m = (((N, H, \mathrm{t}, \Sigma_i, o), \succeq_i), Aw_i)$ with $Aw_i(h) = H$ for all $h \in H$ and $\mathcal{F}(\mathcal{E}^m, h) = \mathcal{E}^m$. Essentially, all players and the modeller are aware of the game and agree on it. Likewise in this section we provide a canonical representation of games with short sight in terms of the general models introduced above (Definitions 13 and 14).

Definition 15 (Canonical representation of short sight) *Take a finite prioritized game with short sight $(\mathcal{G}^P, \mathrm{s})$ where $\mathcal{G}^P = ((N, H, \mathrm{t}, \Sigma_i, o), P_i)$. Let also h be a finite non-terminal history and $\mathcal{E}\lceil_h$ the resulting extensive game as in Definition 2. The* canonical representation *of $(\mathcal{G}^P, \mathrm{s})$ consists of the tuple*

$$\mathcal{E}^{(\mathcal{G}^P, \mathrm{s})} = (\{\{(\mathcal{E}\lceil_h, Aw_i\lceil_h) \mid h \in H\}, \mathcal{E}^m\}, \mathcal{E}^m, \mathcal{F})$$

where: 1) $\mathcal{E}^m = (((N, H, \mathrm{t}, \Sigma_i, o), \succeq_i), Aw_i)$ with $Aw_i(h) = H\lceil_h = \mathrm{s}(h)$; 2) $Aw_i\lceil_h(h') = Aw_i(h, h')$; 3) for each ${}^+\mathcal{E} \in \Gamma$, ${}^+\succeq_i = ({}^+Z \times {}^+Z) \cap \succeq^{P_i}$; 4) $\mathcal{F}(\mathcal{E}^m, h) = (\mathcal{E}\lceil_h, Aw_i\lceil_h)$; 5) $\mathcal{F}((\mathcal{E}\lceil_h, Aw_i\lceil_h), h') = (\mathcal{E}\lceil_{(h,h')}, Aw_i\lceil_{(h,h')})$.

In words, a game with short sight can be represented as a game with awareness where at each choice point players believe to be playing the game induced by their sight. Specifically, the first item says that the modeller knows the structure of the game and the sight of the players at each point. The second item says that players' sight in each augmented game agrees with their sight in the original game. The third item says that every augmented game is consistent with the P-sequence in its terminal nodes. The fourth and fifth item say that the awareness function returns the sight of the players at each decision point.

The following result shows that the above representation yields the right sort of games with awareness.

Theorem 4 *Let $(\mathcal{G}^P, \mathrm{s})$ be a game with short sight. $\mathcal{E}^{(\mathcal{G}^P, \mathrm{s})}$ is a game with awareness.*

Proof. We first need to check that Γ^* is made by a countable set of augmented games and then that they satisfy the axioms given in Definition 14. As for the first part we need to show that the axioms of Definition 12 are satisfied: [A1] we know that the game $(\mathcal{G}^P, \mathrm{s})$ is finite (Definition 15) and that each $\mathcal{E}\lceil_h$ for h being a history of $\mathcal{E}$ is a finite extensive game (Proposition 2); [A2] Aw_i is well defined, as it associates each history of each augmented game exactly the sight of the player who moves at that history. Players' sight is closed under prefixes by Definition 8; [A11-12] Notice that by the construction in Proposition 2 for each history $h \in H$ we have that $\overline{h} = h$. By reflexivity of preferences we obtain the desired result. As for the second part we need to show that the following axioms are satisfied: [M1-M3, C1] Consequence of Definition 15 . $\square$

4.3 Characterization result

In this section we provide the constraints that a game with awareness needs to satisfy in order to be the canonical representation of some game with short sight. Before doing this we introduce the auxiliary notion of game pruning.

Definition 16 (Game pruning) *Let $\mathcal{E} = ((N, H, \mathrm{t}, \Sigma_i, o), \succeq_i)$ be a finite extensive game. The game $\mathcal{E}' = ((N', H', \mathrm{t}', \Sigma_i', o'), \succeq_i')$ is a* pruning *of game $\mathcal{E}$ whenever 1) $N = N'$; 2) $H' \subseteq H$ and H' is a finite set of histories closed under prefixes; 3) for each $h' \in H', \mathrm{t}'(h') = \mathrm{t}(h')$; 4) $\Sigma_i' = \{\sigma_i \in \Sigma_i \mid \sigma_i : h' \to A$ for $h' \in H'$ with $\mathrm{t}'(h') = i$ and there is a $h'' \in H'$ with $h' \lhd h''\}$; 5) for each $\sigma' \in \Sigma'$, $o'(\sigma') = z'$ whenever $z' \in Z'$ and is obtained by executing σ'.*[10]

A game pruning of an extensive game $\mathcal{E}$ is just $\mathcal{E}$ deprived of some histories, preserving the structure of strategies and turn function and defining the outcome function accordingly. Notice that a game pruning of a game is nothing but what we called a sight (recall Definition 8), defined at the root of the game.

The following definition makes use of game prunings, isolating a class of games with awareness with which we will be able to exactly characterize games with short sight.

Definition 17 (Coherence) *Let $\mathcal{E}^{Aw} = (\Gamma, \mathcal{E}^m, \mathcal{F})$ be a game with awareness based on a finite extensive game $\mathcal{E} = ((N, H, \mathrm{t}, \Sigma_i, o), \succeq_i)$. We call $\mathcal{E}^{Aw}$* coherent *if it satisfies the following constraints:*

K1 *the game $\mathcal{E}^m$ is the tuple $(((N, H, \mathrm{t}, \Sigma_i, o), \succeq_i), Aw_i)$ with $Aw_i(h) = H'$ for H' being the set of histories of some game ${}^+\mathcal{E} \in \Gamma$;*

K2 *the set Γ comprises $\mathcal{E}^m$ and for each $h \in H$ a set of $|H|$ augmented games of the form $(\mathcal{E}'|_h, Aw_i')$, with $\mathcal{E}'$ being a* pruning *of $\mathcal{E}$, and $Aw_i'(h') = Aw_i(h, h')$;*

K3 *there exists a total preorder $\succeq_i^H$ on H extending $\succeq_i$ such that for each ${}^+\mathcal{E} \in \Gamma$ we have that ${}^+\succeq_i = \succeq_i^H \cap ({}^+Z \times {}^+Z)$, i.e. histories get the same preferences across augmented games;*

K4 *$\mathcal{F}(\mathcal{E}^m, h') = (\mathcal{E}'|_{h'}, Aw_i')$, for $\mathcal{E}'$ being the pruning of $\mathcal{E}$ associated to h';*

K5 *for each $(\mathcal{E}'|_h, Aw_i') \in \Gamma$ we have that $\mathcal{F}((\mathcal{E}'|_h, Aw_i'), h') = (\mathcal{E}'|_{(h,h')}, Aw_i'')$, where $Aw_i''(h'') = Aw_i(h, h', h'')$.*

The constraints deal with the game form structure and the preferences of coherent games with awareness. Axiom K1 states that the modeller has a perfect view of the game and of the awareness of each player at each history. Notice that by K1, awareness of players agrees at each decision point.[11] Axiom K2 states that players can only see a part of the real game being played. Axiom K3 deals instead with the preference relations and ensures that histories are evaluated according to the same criteria if observed from different points. Axioms K4-5 state that what players believe to be true in the real game at a point coincides with their awareness level at that point. Notice the resemblance of these axioms with the conditions on Definition 15.

We first prove the following lemma:

Proposition 5 (P-sequence existence) *Assume a game with awareness $\mathcal{E}^{Aw} = (\Gamma, \mathcal{E}^m, \mathcal{F})$ that is coherent. We can construct a finite game with short sight $\mathcal{G}^P = (\mathcal{G}, P_i)$ such that $Z \times Z \cap \succeq^{P_i} = \succeq_i$ where Z and $\succeq_i$ are the terminal histories and the preference relation for player i in any ${}^+\mathcal{E} \in \Gamma$.*

Proof (sketch). Let $((N, H, \mathrm{t}, \Sigma_i, o) \succeq_i)$ be the game $\mathcal{E}$ upon which $\mathcal{E}^m$ is based. Consider its game form $(N, H, \mathrm{t}, \Sigma_i, o)$. We construct the desired P-sequence as follows. Let $\succeq_i^H$ be the total preorder required by axiom K3 (Definition 17)

[10]Formally, for $z = (z_1, z_2, \ldots, z_{l(z)})$ and $\forall i \in \{1, 2, \ldots, l(z)\}$ we have that $\sigma_{\mathrm{t}(z_i)}(z_i) = z_{i+1}$.

[11]The requirement looks rather strong, but notice that for decision making purposes the only awareness level that matters is the one of the player who is to move.

and let $\succ_i^H$ indicate its strict counterpart. Let moreover $\mathcal{H} = \{[h] \mid h' \in [h] \iff h' \succeq_i^H h \text{ AND } h \succeq_i^H h'\}$. Intuitively, $\mathcal{H}$ is the set of all equivalence classes induced by the relation $\succeq_i^H$. The desired P-sequence $(\mathcal{H}, \succ)$ is so defined for each $\mathbf{H}, \mathbf{H}' \in \mathcal{H}$: $\mathbf{H} \succ \mathbf{H}'$ if and only if for some $x \in \mathbf{H}, y \in \mathbf{H}'$ we have $x \succ_i^H y$. We need to show (i) that $(\mathcal{H}, \succ)$ is indeed a P-sequence and (ii) that it displays the required properties. As for (i) set $\mathcal{H}$ is clearly a finite set of subsets of H. We are left to show that the relation $\succ$ is (a) irreflexive (b) transitive (c) asymmetric and (d) total. (a) Suppose not, then for some $\mathbf{H} \in \mathcal{H}$ and $x, y \in \mathbf{H}$ we would have $x \succ_i^H y$, leading to contradiction. Claims (b) - (c) - (d) can be proven by a similar procedure. (ii) For any two histories h', h and ${}^+\mathcal{E} \in \Gamma$ with preference relation $\succeq_i$ and with h', h among the terminal histories of ${}^+\mathcal{E}$ we need to show that: $h \succeq_i h'$ if and only if $h \succeq^{(\mathcal{H},\succ)} h'$. Both directions are straightforward. $\Box$

We are now ready to formulate our main result.

Theorem 6 (Correspondence) *Let $\mathcal{E}^{Aw} = (\Gamma, \mathcal{E}^m, \mathcal{F})$ be a coherent game with awareness based on $\mathcal{E}$. There exists a finite game with short sight $(\mathcal{G}^P, \mathrm{s})$ such that its canonical representation $\mathcal{E}^{(\mathcal{G}^P,\mathrm{s})}$ is such that $\mathcal{E}^{Aw} = \mathcal{E}^{(\mathcal{G}^P,\mathrm{s})}$.*

PROOF. We proceed by construction. Let $((N, H, \mathrm{t}, \Sigma_i, o) \succeq_i)$ be the game $\mathcal{E}$. Consider its game form $(N, H, \mathrm{t}, \Sigma_i, o)$. To construct the game $(\mathcal{G}^P, \mathrm{s})$ first use Proposition 5 to obtain the desired P-sequence P_i for each player. As for the sight function we simply impose the following: for every history $h \in H$, and every player $i \in N$ we have that $\mathrm{s}(h) = Aw_i(h)$, where $Aw_i(h) = H'$ is the awareness function as appears in $\mathcal{E}^m$. The requirements of Definition 8 are satisfied as a consequence of the fact that $\mathrm{s}(h)$ is always the set of histories of some finite game following h (Definition 17). Now the fact that $\mathcal{E}^{Aw} = \mathcal{E}^{(\mathcal{G}^P,\mathrm{s})}$ follows from Definitions 15 and 17. $\Box$

Theorems 4 and 6 have established a precise link between the most general class of games with awareness introduced in [4]—i.e., games with awareness and lack of common knowledge of the game structure—and the class of games with short sight, namely that the latter is a special subclass of the former. This puts the results presented in Section 3 in an interesting light. In fact, [4] did not develop any equilibrium analysis of games with awareness and lack of common knowledge of the game structure. The notion of sight compatible subgame perfect equilibrium can therefore be viewed as a first principled generalization of subgame perfection to a specific form of unawareness—short sight.

5. CONCLUSIONS

Inspired by Joseph Halpern's invited talk at AAMAS 2011—*Beyond Nash-Equilibrium: Solution Concepts for the 21st Century*—and moving from simple considerations concerning real life game playing (Example 1), the paper has proposed a class of games where players are characterized by two key features: 1) they have only partial access to the game structure including, critically, having possibly no access to terminal nodes; 2) they play according to extrinsic evaluation criteria, which have here been modeled as sequences of properties of histories (Definition 3). The paper has shown thas such games 1) always possess an appropriate refinement of the subgame perfect equilibrium concept (Theorem 3); 2) are an interesting—because of the above equilibrium properties—subclass of the most general class of games with awareness proposed by Halpern and Rêgo (Theorems 4 and 6) which, although introduced in [4], had not yet been object of investigation from the point of equilibrium analysis.

Future work will focus on weakening two assumptions. First, the fact that in solving games with short sight we have presupposed that players only consider their own sight (Definition 10) and that the evaluative components of the game—the P-sequences—are common knowledge. Dropping these assumptions could open up interesting avenues of research concerning learning methods by means of which players could infer other players' evaluation criteria and sights, i.e., other players' types. This would bring the game theoretical method of equilibrium analysis close to established game-playing techniques in artificial intelligence and some of its recent developments such as the theory of *general game playing* [3]. Second, it is clear that the granularity of their evaluation criteria has direct impact on players' performance in a game with short sight. We have currently defined P-sequences as sequences of sets of histories. A more refined approach would take into consideration the (formal) language by means of which players express their evaluation criteria. Methods from logic could then be used to compare the expressivity of different languages for P-sequences, possibly correlating such expressivity to players' performance in the games.

Acknowledgements. Paolo Turrini acknowledges the support of the National Research Fund of Luxembourg for the Trust Games project (1196394), cofunded under the Marie Curie Actions of the European Commission (FP7-COFUND). The authors would like to thank the anonymous reviewers of AAMAS'12 for the helpful suggestions.

6. REFERENCES

[1] Y. Feinberg. Subjective reasoning—games with unawareness. Technical Report Research Paper Series 1875, Stanford Graduate School of Business, 2004.

[2] Y. Feinberg. Games with incomplete unawareness. Technical report research paper series, Stanford Graduate School of Business, 2005.

[3] M. Genesareth, N. Love, and B. Pell. General game playing: Overview of the aaai competition. *AAAI Magazine*, 2005.

[4] J. Y. Halpern and L. C. Rêgo. Extensive games with possibly unaware players. In *In Proceedings of the 5th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2006)*, pages 744–751, 2006.

[5] J. Y. Halpern and L. C. Rêgo. Extensive games with possibly unaware players. *CoRR*, abs/0704.2014, 2007.

[6] F. Liu. A two-level perspective on preference. *Journal of Philosophical Logic*, 40(3):421–439, 2011.

[7] M. Osborne and A. Rubinstein. *A course in Game Theory.* The MIT Press, 1994.

[8] L. C. Rêgo and J. Y. Halpern. Generalized solution concepts in games with possibly unaware players. *International Journal of Game Theory*, (to appear).

[9] J. Watson. *Secrets of Modern Chess Strategy: Advances since Nimzowitch.* Gambit Publications, 1998.

New Results on the Verification of Nash Refinements for Extensive–Form Games

Nicola Gatti
Politecnico di Milano
Piazza Leonardo da Vinci 32
Milano, Italy
ngatti@elet.polimi.it

Fabio Panozzo
Politecnico di Milano
Piazza Leonardo da Vinci 32
Milano, Italy
panozzo@elet.polimi.it

ABSTRACT

The computational study of strategic interaction situations has recently deserved a lot of attention in multi–agent systems. A number of results on strategic–form games and zero–sum extensive–form games are known in the literature, while general–sum extensive–form games are not studied in depth. We focus on the problem to decide whether or not a solution is a refinement of the Nash equilibrium (NE) for extensive–form games. Refinements are needed because the NE concept is not satisfactory for this game class. While verifying whether a solution is an NE is in $\mathcal{P}$, verifying whether it is a NE refinement may be not (all the results known so far show $\mathcal{NP}$–hardness). In this paper, we provide the first positive result, showing that verifying a *sequential equilibrium* with any number of agents and a *quasi perfect equilibrium* with two agents are in $\mathcal{P}$. We show also that when the input is expressed in (non–perturbed) sequence form even the problem to verify a subgame perfect equilibrium is $\mathcal{NP}$–complete and that sequence form, if applicable, must be rethought to verify (and therefore to compute) an extensive–form perfect equilibrium.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Distributed Artificial

General Terms

Algorithms, Economics

Keywords

Game Theory (cooperative and non-cooperative)

1. INTRODUCTION

The study of formal methods for addressing strategic interaction problems among rational agents has recently received an increasing attention in artificial intelligence and, especially, in the multi–agent system community. The aim is the development of algorithms to automate software agents and robots. Formal methods can allow one to model situations and define what is the optimal behavior an agent can have. Game theory and microeconomics represent the most elegant formal methods for strategic interaction scenarios [8]. Customarily, a scenario is modeled as a *game* in which one distinguishes the *mechanism*, defining the rules of the game (i.e., number of agents, actions available to the agents, game sequential structure, outcomes, and agents' preferences over the outcomes), from the *strategies*, defining how each single agent behaves at every decision node she acts. A *solution* of a game is strategy that is stable according to some solution concept. The basic solution concept is the Nash equilibrium (NE) constraining the strategy of each agent to be optimal given the strategies of the others.

Game theory and microeconomics provide only models and solution concepts, but they do not provide computational tools to deal with games. The development of these tools is an interesting topic, with the name of *equilibrium computation*, in computer science. The main open problems are, e.g., the verification that a solution is a given solution concept and the search for some exact or approximate solution concept. A number of computational results are known on the NE, we cite a few. Computing an exact NE [6, 7] and approximating it [4] are $\mathcal{PPAD}$–complete. $\mathcal{PPAD}$ is in $\mathcal{NP}$, it does not include $\mathcal{NP}$–complete problems unless $\mathcal{NP}$ = co–$\mathcal{NP}$, and it is not known whether $\mathcal{PPAD}$ is in $\mathcal{P}$, but it is commonly believed that it is not. Instead, the problem to verify whether or not a solution is a NE is in $\mathcal{P}$.

A number of works deal with the problem to compute a NE with general–sum strategic–form games (especially with two agents), e.g., [2, 19, 21, 22], and with the problem to solve large zero–sum extensive–form games, e.g., [11, 12]. The problem to study general–sum extensive–form games has received less attention and appears as one of the "next issues of the agenda" according to [28]. With these games, the NE is not satisfactory and refinements are needed. The most common refinements are [25]: the *subgame perfect equilibrium* (SPE) when information is perfect and the *sequential equilibrium* (SE), *quasi perfect equilibrium* (QPE), and *extensive–form perfect equilibrium* (EFPE) when information is imperfect. While the SE is the "natural" extension of the SPE to the case with imperfect information, perfect equilibria (both QPE and EFPE) pose more severe constraints, requiring the strategies to be optimal also when the agents tremble over non–optimal strategies. QPEs and EFPEs differentiate as follows: in a QPE each agent does not consider her own trembles, while in EFPEs she does.

While the verification of an NE is easy, few results are known about the verification of NE refinements for extensive–form games. The verification problem is of extraordinary

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

importance, allowing an user to verify whether a software agent is an optimizer or not. In the case this problem is intractable, we cannot certificate that the behavior of an agent is optimal and therefore the use of autonomous agents appears impractical. This would push one to resort to new approximate solution concepts. The unique results known so far in the literature are negative. More precisely, verifying whether a solution is a QPE or an EFPE is $\mathcal{NP}$–hard with three or more agents [14]. For SEs it is known only an algorithm that can be exponential in the worst case [15].

In the present paper, we provide new contributions on the NE refinement verification. More precisely, we provide two prominent positive results: both problems of verifying a SE with an arbitrary number of agents and a QPE with two agents are in $\mathcal{P}$. This supports the employment of these solution concepts in practice. In addition, we provide two negative results. The first result shows that, when the input is expressed in (non–perturbed) sequence form [27], even verifying an SPE is $\mathcal{NP}$–complete. The second result shows that the sequence form, if applicable, must be rethought to verify (and compute [9]) an EFPE with two agents (if not applicable, verifying an EFPE requires non–linear optimization and therefore the problem is not probably in $\mathcal{P}$).

2. EXTENSIVE–FORM GAMES AND EQUILIBRIUM COMPUTATION

2.1 Game definition and strategies

A *perfect–information* extensive–form game [8] is a tuple $(N, A, V, T, \iota, \rho, \chi, u)$, where: N is the set of agents ($i \in N$ denotes a generic agent), A is the set of actions ($A_i \subseteq A$ denotes the set of actions of agent i and $a \in A$ denotes a generic action), V is the set of decision nodes ($V_i \subseteq V$ denotes the set of decision nodes of i), T is the set of terminal nodes ($w \in V \cup T$ denotes a generic node and w_0 is root node), $\iota : V \to N$ returns the agent that acts at a given decision node, $\rho : V \to \wp(A)$ returns the actions available to agent $\iota(w)$ at w, $\chi : V \times A \to V \cup T$ assigns the next (decision or terminal) node to each pair w, a where a is available at w, and $u = (u_1, \dots, u_n)$ is the set of agents' utility functions $u_i : T \to \mathbb{R}$. Games with *imperfect information* extend those with perfect information, allowing one to capture situations in which some agent cannot observe some action undertaken by the other agents. We denote by $V_{i,h}$ the h–th *information set* of agent i. An information set is a set of decision nodes such that when an agent plays at one of its nodes she cannot distinguish the node in which she is playing. For the sake of simplicity, we assume that every information set has a different index h, thus we can univocally identify an information set by h. An imperfect–information game is a tuple $(N, A, V, T, \iota, \rho, \chi, u, H)$ where $(N, A, V, T, \iota, \rho, \chi, u)$ is a perfect–information game and $H = (H_1, \dots, H_n)$ induces a partition $V_i = \bigcup_{h \in H_i} V_{i,h}$ such that for all $w, w' \in V_{i,h}$ we have $\rho(w) = \rho(w')$. We focus on games with *perfect recall* where each agent recalls all her previous actions and her previous observations. Perfect recall poses severe constraints over the structure of the information sets, we omit their description here, not being necessary for our work, and point an interested reader to [8].

There are three representations for extensive–form games: *normal form* [26], *agent form* [17, 23], and *sequence form* [27]. In this paper, we resort to the agent and sequence forms.

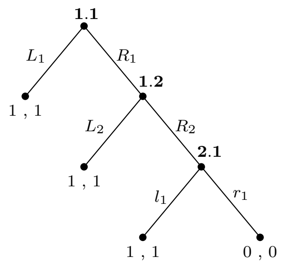


Figure 1: Example of two–agent perfect–information extensive–form game.

In the agent form, it is assumed that at each information set a different agent plays (e.g., in Fig. 1 there are three different agents, one per information set). In this way, a strategy (said *behavioral*) is represented as a probability distribution over the actions available at each single information set independently of the probability with which such an information set is reached. A behavioral strategy profile is $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_{|N|})$ where σ_i is the strategy of agent i. We denote by $\sigma_{i,a}$ the probability associated with action $a \in A_i$.

In the sequence form, a strategy is represented as a probability distribution over *sequences*. A sequence $q \in Q_i$ is a set of consecutive actions $a \in A_i$, where $Q_i \subseteq Q$ is the set of sequences of agent i and Q is the set of all the sequences. A sequence can be *terminal*, if, combined with some sequence of the opponents, it leads to a terminal node, or *non–terminal*, if it cannot lead to any terminal node for every opponents' sequence. In addition, the initial sequence of every agent, denoted by q_0, is said *empty sequence* and, given sequence $q \in Q_i$ leading to some information set $h \in H_i$, we say that q' *extends* q (denoted by $q' = q|a$) if the last action a' of q' (denoted by $a(q)$) belongs to $\rho(w)$ with $w \in V_{i,h}$. We denote a sequence–form strategy profile as a vector by $\mathbf{x} = [\mathbf{x}_1, \dots, \mathbf{x}_{|N|}]$ where $\mathbf{x}_i$ is the strategy of agent i and we denote by $x_{i,q}$ the probability associated with sequence $q \in Q_i$. Well defined strategies are such that, for every information set $h \in H_i$, the probability $x_{i,q}$ assigned to the sequence q leading to h is equal to the sum of the probabilities $x_{i,q'}$s where q' extends q at h. Sequence form constraints can be conveniently described as $F_i \mathbf{x}_i = \mathbf{f}_i$, where F_i is an opportune matrix and $\mathbf{f}_i$ is an opportune vector. The agent i's utility is represented as a sparse multi–dimensional array, denoted by U_i, specifying the value associated to every combination of terminal sequences of all the agents. The size of the sequence form representation is linear in the size of the game tree.

A sequence–form strategy $\mathbf{x}_i$ is equivalent to a number (precisely, a compact set) of behavioral strategies σ_i and the relationship is non–linear. More precisely, given an information set $h \in H_i$ and called $q \in Q_i$ the sequence leading to h, the behavioral strategy $\sigma_{i,a}$ related to the actions $a \in \rho(w)$ with $w \in V_{i,h}$ and $q' = q|a$ is $\sigma_{i,a(q')} = \frac{x_{i,q'}}{x_{i,q}}$ if $x_{i,q} > 0$ and 0 otherwise. The two representations have different degrees of expressiveness, e.g., sequence–form strategies, differently from behavioral ones, do not specify the actions that would be played at information sets reached with zero probability.

Several solution concepts (see below) are based on the idea of *perturbed strategies*. Call $l_{i,a}(\epsilon) > 0$ the *perturbation* (in terms of probability) over action $a \in A_i$ such that $\lim_{\epsilon \to 0} l_{i,a}(\epsilon) = 0$ and ϵ is a positive value. We denote by $\mathbf{l}_i(\epsilon)$ the vectors of the perturbations over all the agent i's actions. A perturbed behavioral strategy profile $\boldsymbol{\sigma}(\epsilon)$ of $\boldsymbol{\sigma}$ is a fully mixed strategy where $\sigma_{i,a} \geq l_{i,a}(\epsilon)$ for all $a \in A_i$ and $\lim_{\epsilon \to 0} \boldsymbol{\sigma}(\epsilon) = \boldsymbol{\sigma}$. Analogously, the idea of perturbation can be applied to the sequence form. In this case, we denote by $\mathbf{x}_i(\epsilon)$ the perturbed sequence form strategy and by $x_{i,q}(\epsilon)$ the perturbed strategy over $q \in Q_i$. The result in [1] shows that we can deal with perturbations $l_{i,a}(\epsilon)$ defined as polynomials in ϵ keeping ϵ a symbolic parameter by resorting to the concept of lexico positiveness (see Appendix A). In our work, we denote by $\mathbf{x}_i(\epsilon^k)$ the coefficients of ϵ^k in $\mathbf{x}_i(\epsilon)$ and $x_{i,q}(\epsilon^k)$ the coefficients of ϵ^k in $x_{i,q}(\epsilon)$.

2.2 Solution concepts

It is well known that the concept of NE is not satisfactory for extensive–form games, allowing agents to play non–credible threats. The concept of SPE refines the concept of NE, constraining a strategy profile to be a NE in every subgame [8], where a subgame is a portion of the game tree defined as follows: it has a root and for every node $w \in V_{i,h}$ belonging to the subgame the whole information set $V_{i,h}$ belongs to the subgame. (A SPE can be easily found by applying *backward induction* [8].) The concept of SPE is satisfactory with perfect–information games, while it is not when information is imperfect. The "natural" extension of the SPE to situations with imperfect information is the SE [16]. We denote by $\mu_i = (\mu_{i,w})$ for every $w \in V_i$ the *beliefs* of agent i where $\mu_{i,w}$ is the probability with which agent i believes to be at node $w \in V_{i,h}$ when she plays at information set h. We denote by $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_{|N|})$ the profile of beliefs. An *assessment* is a pair $(\boldsymbol{\mu}, \boldsymbol{\sigma})$. An SE is an assessment $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ such that: every σ_i is *sequentially optimal* (in the sense of backward induction) with respect to μ_i, and every μ_i is *consistent* (in the sense of Kreps and Wilson) with respect to σ_{-i}. Consistency of $\boldsymbol{\mu}$ with respect to $\boldsymbol{\sigma}$ requires that there exists a perturbed strategy profile $\boldsymbol{\sigma}(\epsilon)$ of $\boldsymbol{\sigma}$ such that, if $\boldsymbol{\mu}(\epsilon)$ the sequence of beliefs derived from $\boldsymbol{\sigma}(\epsilon)$ by using the Bayes rule, $\lim_{\epsilon \to 0} \boldsymbol{\mu}(\epsilon) = \boldsymbol{\mu}$. With perfect information every SPE is also an SE and *vice versa*. Instead, when information is imperfect, the SEs constitute a subset of the SPEs.

Example 2.1. *Consider the game in Fig. 1. The pure strategy SEs (and SPEs) are:* $(\sigma_{1,L_1} = 1, \sigma_{1,L_2} = 1, \sigma_{2,l_1} = 1)$, $(\sigma_{1,L_1} = 1, \sigma_{1,R_2} = 1, \sigma_{2,l_1} = 1)$, $(\sigma_{1,R_1} = 1, \sigma_{1,L_2} = 1, \sigma_{2,l_1} = 1)$, $(\sigma_{1,R_1} = 1, \sigma_{1,R_2} = 1, \sigma_{2,l_1} = 1)$.

The idea of *perfection*, introduced by Selten in [23], is strictly correlated to the idea of perturbed strategy. Basically, a strategy profile is perfect when it is optimal even with perturbations over the strategies. The rationale behind perturbations is that agents do not perfectly play their optimal strategy, but they tremble with a very small probability over non–optimal strategies. The application of perturbation to the three (normal, agent, sequence) forms of a game may lead to different concepts of equilibria.

A strategy profile $\boldsymbol{\sigma}$ is a QPE if there exists a perturbed strategy profile $\boldsymbol{\sigma}(\epsilon)$ of $\boldsymbol{\sigma}$ such that $\sigma_{i,a}(\epsilon) \geq l_{i,a}(\epsilon)$ and every σ_i is a best response to $\sigma_{-i}(\epsilon)$ for every $\epsilon \leq \overline{\epsilon}$ for some $\overline{\epsilon} > 0$ [24]. In a QPE every agent takes into account the opponents' trembles, but not own. For every combination of $\mathbf{l}_i(\epsilon)$ there is a potentially different QPE. The authors show in [20] that quasi perfection can be captured by using a specific class of perturbations with the sequence form constraining that for every pair of sequences $q, q' \in Q_i$ with $q = q'|a(q)$ the minimum degree of k such that $l_{i,q}(\epsilon^k)$ is strictly positive is strictly smaller than the minimum degree of k such that $l_{i,q'}(\epsilon^k)$ is strictly positive, formally, $\min_{l_{i,q}(\epsilon^k)>0} \{k\} > \min_{l_{i,q'}(\epsilon^k)>0} \{k\}$.

Other solution concepts are the *normal–form perfect equilibrium* (NFPE; when perturbations are over normal–form strategies, but it is not a satisfactory solution concept) and the *extensive–form perfect equilibrium* (EFPE; it is defined as the QPE except that an agent takes into account her own trembles in addition to those of the opponents).

Example 2.2. *Consider the game represented in Fig. 1. The pure strategy QPEs are:* $(\sigma_{1,L_1} = 1, \sigma_{1,L_2} = 1, \sigma_{2,l_1} = 1), (\sigma_{1,R_1} = 1, \sigma_{1,L_2} = 1, \sigma_{2,l_1} = 1)$. *Notice that* $(\sigma_{1,R_1} = 1, \sigma_{1,R_2} = 1, \sigma_{2,l_2} = 1)$ $(\sigma_{1,R_1} = 1, \sigma_{1,R_2} = 1, \sigma_{2,l_1} = 1)$, *that is an SE, is not a QPE. This is because, accounting for any perturbed* $\sigma_{2,l_1}(\epsilon)$*, the utility expected by agent 1 from making action* R_2 *(i.e.,* $\sigma_{2,l_1}(\epsilon) < 1$*) is strictly smaller than the utility she expects from making action* L_2 *(i.e., 1). The unique EFPE, when agents account for own trembles, is* $(\sigma_{1,L_1} = 1, \sigma_{1,L_2} = 1, \sigma_{2,l_1} = 1)$.

2.3 Known computational results

The sequence form is the most efficient representation to compute an NE (normal form is exponentially larger, while agent form poses highly non–linear constraints over the agents' best response optimization problems). The main results on the computation of an NE are with two agents. The problem to search for an NE is formulated as a linear–complementarity problem (LCP) and solved by employing the Lemke's algorithm [18], a generalization of the Lemke–Howson algorithm [19]. The problem to verify whether a strategy profile, both in sequence form and agent form (in this case deriving the corresponding sequence form strategies), is an NE can be easily solved in polynomial time by checking whether or not the constraints are satisfied.

The computation and verification problems for an SE are open [14] and it is not known whether it is possible to address them in sequence form or, as it is commonly believed, it is necessary the agent form. The unique result on the verification of an SE is provided in [15]. They propose finite–step algorithm to verify whether an assessment is an SE, but, as they state it, the number of steps accomplished by the algorithm can be exponential in the worst case. A slightly different problem is studied in [14], where the authors show in that with three or more agents, verifying whether there is an SE with a given strategy is $\mathcal{NP}$–hard.

In [20] the authors use the Lemke's algorithm applied to the sequence form with perturbations $\mathbf{l}_1(\epsilon)$, $\mathbf{l}_2(\epsilon)$ with $l_{i,q}(\epsilon) = \epsilon^{|q|}$ where $|q|$ is the length of sequence q to compute a QPE when agents are two (details are in Appendix B). This places that such a problem in the $\mathcal{PPAD}$ class. Instead, the verification problem is currently open with two agents. Differently from the verification of an NE, verifying whether a strategy profile is a QPE is a search problem in which perturbations $\mathbf{l}_1(\epsilon), \mathbf{l}_2(\epsilon)$ need to be found to satisfy the QPE constraints. With three or more agents the verification problem is shown to be $\mathcal{NP}$–hard [14].

Other known results on the equilibrium verification problem are: the verification of a NFPE with two agents is in $\mathcal{P}$, while the verification of a NFPE and of an EFPE with three or more agents is $\mathcal{NP}$–hard [14]. No result is known for EFPE with two agents and it is not known even whether or not the sequence form can be employed.

3. VERIFICATION WITH AGENT FORM

We report a positive (tractable) result on the verification of an SE by providing an algorithm that works with the agent form. This is possible since we do not need to use perturbations. When instead perturbations must be considered, as for the verification of a QPE, working with the agent form appears hard since the verification problem is equivalent to the problem to search for an appropriate perturbation over the behavioral strategies and this problem is highly non–linear because the perturbations at different information sets would be multiplied. We state the following theorem, whose proof provides a polynomial time algorithm based on linear programming.

Theorem 3.1. *Given a game with an arbitrary number of agents, it is in $\mathcal{P}$ the problem to decide whether or not an assessment $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ is an SE.*

Proof. This decision problem requires one to verify two correlated properties: sequential rationality and consistency. Sequential rationality of $\boldsymbol{\sigma}$ can be easily verified by backward induction on the basis of $\boldsymbol{\mu}$. This task requires a number of maximizations that is linear in the size of the game, and each single maximization is over a number of actions that is linear in the size of the game. Verifying consistency of $\boldsymbol{\mu}$ is an harder task. By definition, it requires one to find a fully mixed perturbed strategy profile $\boldsymbol{\sigma}(\epsilon)$ such that the beliefs $\boldsymbol{\mu}(\epsilon)$ derived from $\boldsymbol{\sigma}(\epsilon)$ by Bayes rule converges to $\boldsymbol{\mu}$ as $\epsilon \to 0$.

The problem to find a $\boldsymbol{\sigma}(\epsilon)$ can be solved by resorting to the concept of *b–labeling* provided by Kreps and Wilson in [16]. A b–labeling for an assessment $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ is a function $\lambda : A \to \mathbb{N}$ that assigns a label (expressed as a non–negative integer number) to all the actions $a \in A$ such that:

$$\begin{array}{lll} \lambda_a = 0 & \iff \sigma_{i,a} > 0 & \forall a \in A_i, i \in N \\ \sum_{a \to w} \lambda_a = \arg\min_{w'} \sum_{a \to w'} \lambda_a & \iff \mu_{i,w} > 0 & \forall w, w' \in V_{i,h}, i \in N, h \in H_i \end{array}$$

We use the symbol '$a \to w$' to denote all the actions $a \in A$ leading to node w from the root node w_0. Given a b–labeling, we can define a fully mixed strategy profile $\overline{\boldsymbol{\sigma}}(\epsilon)$ as:

$$\overline{\sigma}_{i,a}(\epsilon) = \begin{cases} c(\epsilon, h, a) \cdot \sigma_{i,a} & \text{if } \sigma_{i,a} > 0 \\ c(\epsilon, h, a) \cdot \epsilon^{\lambda_a} & \text{otherwise} \end{cases}$$

where a is an action played by some agent at information set h, and $c(\epsilon, h, a)$ is the appropriate normalizing constant. Kreps and Wilson proved that $\boldsymbol{\mu}$ is consistent to $\boldsymbol{\sigma}$ if and only if the above $\overline{\boldsymbol{\sigma}}(\epsilon)$ is well defined (i.e., b–labeling exists). We show below that the problem to search for a b–labeling can be accomplished in polynomial time.

The bottom line of proof is the following. First, we formulate the problem to find a b–labeling as a linear integer mathematical program [29], second, we show that the coefficient matrix associated with the mathematical program in standard form is *totally unimodular* [3] and the right hand is integer. Therefore, the integer mathematical program can be solved in polynomial time, all the basic solutions of the relaxed continuous mathematical program being integer.

The integer mathematical programming formulation is (γ and ν denote auxiliary variables, while s and t denote slack variables):

$$\begin{array}{rll} \min \sum_{a \in A} \lambda_a & & (1) \\ \lambda_a = 0 & \forall a \in A, \sigma_{i,a} > 0, i \in N & (2) \\ \lambda_a - s_a = 1 & \forall a \in A, \sigma_{i,a} = 0, i \in N & (3) \\ \gamma_{w_0} = 0 & & (4) \\ \gamma_{w'} + \lambda_a - \gamma_w = 0 & \forall w, w' \in V, a \in A, w = \chi(w', a) & (5) \\ \gamma_w - \nu_h = 0 & \forall h \in H_i, w \in V_{i,h}, i \in N, \mu_{i,w} > 0 & (6) \\ \gamma_w - \nu_h - t_n = 1 & \forall h \in V_{i,h}, i \in N, \mu_{i,w} = 0 & (7) \\ \lambda_a \in \mathbb{N} & \forall a \in A & (8) \\ s_a \geq 0 & \forall a \in A & (9) \\ t_w \geq 0 & \forall w \in V & (10) \end{array}$$

Constraints (2) force labels λ_a of actions a played with positive probability to be equal to zero; constraints (3) force labels λ_a of actions a played with zero probability to be at least one; constraint (4) assigns a value of zero to auxiliary variable γ_{w_0} associated with root node w_0; constraints (5) assign the auxiliary variable γ_w associated with node w a value equal to the sum of the value of the parent node w' and the label of the action connecting w' to w; constraints (6) force the values of all the γ_ws associated with the nodes ws with $\mu_{i,w} > 0$ belonging to the same information set to be same (i.e., ν_h); constraints (7) force the other nodes w (those with $\mu_{i,w} = 0$) to have a value γ_w strictly larger than the minimum value of the information set (i.e., ν_h); constraints (8)–(10) fix the domains of the variables (notice that, with these domains, all the variables have non–negative values).

The above constraints can be expressed as $M\mathbf{y} = \mathbf{b}$ with $\mathbf{y} \geq 0$ and $\boldsymbol{\lambda}$ constrained to have non–negative integer values, where:

$$M = \left[\begin{array}{c|c|c|c|c} C & 0 & 0 & 0 & 0 \\ \hline C' & 0 & 0 & -I & 0 \\ \hline D & E & 0 & 0 & 0 \\ \hline 0 & G & K & 0 & 0 \\ \hline 0 & G' & K' & 0 & -I \end{array}\right], \mathbf{y} = \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\gamma} \\ \boldsymbol{\nu} \\ \mathbf{s} \\ \mathbf{t} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{1} \end{bmatrix}$$

such that $C\boldsymbol{\lambda} = 0$ codes constraints (2), $C'\boldsymbol{\lambda} - I\mathbf{s} = 1$ codes constraints (3), $D\boldsymbol{\lambda} + E\boldsymbol{\gamma} = 0$ codes constraints (4) and (5), $G\boldsymbol{\gamma} + K\boldsymbol{\nu} = 0$ codes constraints (6), and $G'\boldsymbol{\gamma} + K'\boldsymbol{\nu} - I\mathbf{t} = 0$ codes constraints (7). The above submatrices have the following properties: C, C', G, and G' have one 1 per row and zero or one 1 per column; D is composed of a row of zero and identity matrix I; E has one '1' in the first row and one '1' and one '−1' in all the other rows; K presents one '−1' per row.

Given that a matrix M is totally unimodular if and only if the transpose M^T is totally unimodular [3], we can restate the theorem of Ghoulia–Houri [10] as: M is totally unimodular if and only if for every subset M' of columns of M it is possible to find a partition of columns $\{M'_1, M'_2\}$ such that (call m'_{kj} a generic element of matrices M'_1 and M'_2):

$$\forall k \left(\sum_{j, m'_{kj} \in M'_1} m'_{kj} - \sum_{j, m'_{kj} \in M'_2} m'_{kj} \right) \in \{-1, 0, 1\} \qquad (11)$$

To prove that this condition holds for M, call Λ_i the k–th block of rows of M (from the top to the bottom) and call Δ_j the j–th block of columns of M (from the left to the right).

At first, we notice that we can remove the last two blocks of columns Δ_4 and Δ_5. Indeed, if constraints (11) are satisfied limiting to the first three blocks of columns (i.e., considering only the elements m'_{ij} belonging to $M'_k \cap \{\Delta_1 \cup \Delta_2 \cup$

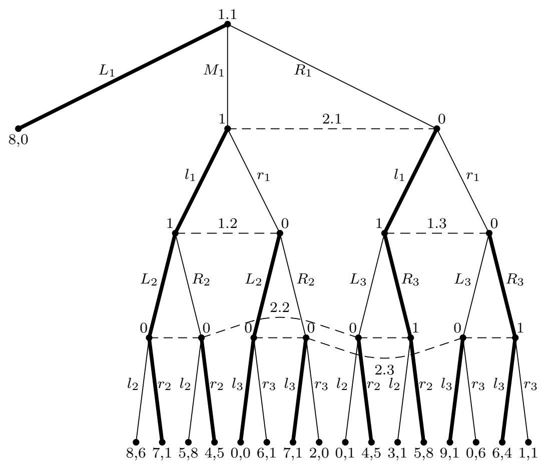


Figure 2: Example of assessment $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ where $\boldsymbol{\sigma}$ is sequentially rational, but $\boldsymbol{\mu}$ is not consistent ($\boldsymbol{\sigma}$ is represented by using bold lines to denote actions played with positive probability and $\boldsymbol{\mu}$ is represented reporting the beliefs close to the nodes of each information set). No b–labeling exists because the constraints due to information set $h = \mathbf{2.1}$ (i.e., $\lambda_{R_1} \geq \lambda_{M_1} + 1$) and due to information set $h = \mathbf{2.2}$ (i.e., $\lambda_{M_1} \geq \lambda_{R_1} + 1$) cannot be satisfied simultaneously.

$\Delta_3\}$), then we can always put columns of M' belonging to Δ_4 or Δ_5 into M'_1 or M'_2 to make constraints (11) satisfied along all the columns. We build M'_1 and M'_2 as follows. Put all the columns of M' belonging to Δ_2 or Δ_3 into M'_1. It can be easily seen that constraints (11) for the rows belonging to Λ_4 and Λ_5 are satisfied (the sum of elements belongs to $\{-1, 0, 1\}$) independently of whether the columns of M' belonging to Δ_1 are put into M'_1 or M'_2. Consider the columns of M'_1 belonging to Δ_2: the sum of the elements of rows belonging to Λ_3 can be $\{-1, 0, 1\}$. It can be easily seen that, D having no more than one '1' per column, we can always put the columns of Δ_1 into M'_1 or M'_2 to make constraints (11) satisfied along the rows belonging to Λ_3. Finally, we observe that constraints (11) are always satisfied along the rows belonging to Λ_1 and Λ_2. Thus, M is totally unimodular and, $\mathbf{b}$ being integer, a b–labeling, if it exists, can be found by linear (continuous) mathematical programming. □

We provide two examples to which we apply the algorithm discussed in the proof of Theorem 3.1.

EXAMPLE 3.2. *Consider the game depicted in Fig. 2 and the assessment $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ where $\boldsymbol{\sigma} = (\sigma_{1,L_1} = 1, \sigma_{1,L_2} = 1, \sigma_{1,R_3} = 1, \sigma_{2,l_1} = 1, \sigma_{2,r_2} = 1, \sigma_{2,l_3} = 1)$ and beliefs $\boldsymbol{\mu}$ are reported in the figure aside the corresponding nodes. No b–labeling exists because the constraints due to information set $h = \mathbf{2.1}$ (i.e., $\lambda_{R_1} \geq \lambda_{M_1} + 1$) and due to information set $h = \mathbf{2.2}$ (i.e., $\lambda_{M_1} \geq \lambda_{R_1} + 1$) cannot be satisfied simultaneously. Therefore, the assessment is not an SE.*

EXAMPLE 3.3. *Consider the game depicted in Fig. 3 and the assessment $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ where $\boldsymbol{\sigma} = (\sigma_{1,M_1} = 1, \sigma_{1,L_2} = 1, \sigma_{1,R_3} = 1, \sigma_{2,l_1} = 1, \sigma_{2,l_2} = 1, \sigma_{2,r_3} = 1)$ and beliefs $\boldsymbol{\mu}$ are reported in the figure aside the corresponding nodes. A b–labeling is: $\lambda_a = 1$ for all $a \in A_i$ with $\sigma_{i,a} = 0$. Therefore, the assessment is an SE.*

4. VERIFICATION WITH SEQUENCE FORM

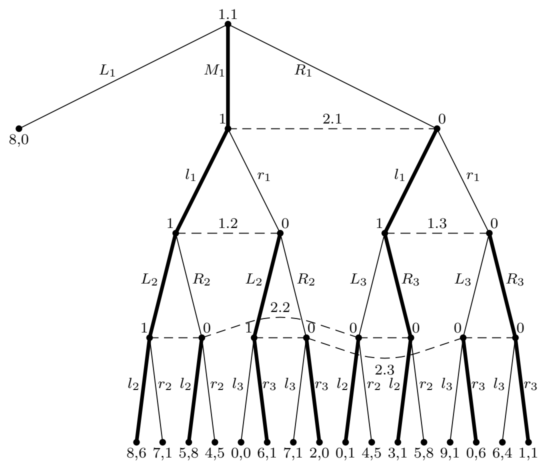


Figure 3: Example of assessment $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ where $\boldsymbol{\sigma}$ is sequentially rational and $\boldsymbol{\mu}$ is consistent ($\boldsymbol{\sigma}$ is represented by using bold lines to denote actions played with positive probability and $\boldsymbol{\mu}$ is represented reporting the beliefs close to the nodes of each information set). The b–labeling is: $\lambda_a = 1$ for all $a \in A_i$ with $\sigma_{i,a} = 0$.

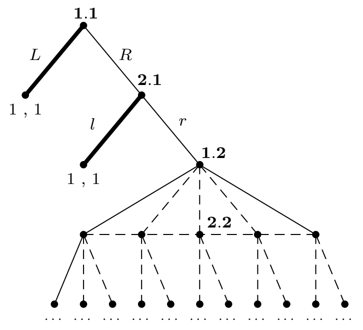


Figure 4: Game used in the proof of Theorem 4.1.

We provide some verification results when we use the sequence form. Initially, we report a negative result even for the SPE when the input strategies of the verification problem are expressed in (non–perturbed) sequence form.

THEOREM 4.1. *Given a game with two agents, it is $\mathcal{NP}$–complete the problem to decide whether or not non–fully perturbed strategies $\mathbf{x}_1, \mathbf{x}_2$ constitute an SPE.*

Proof. We reduce this to the problem to decide whether there is a NE with some property. This problem was shown $\mathcal{NP}$–complete in [5]. The reduction is based on the game tree depicted in Fig. 4. The subgame starting with information set $h = \mathbf{1.2}$ and including information set $h = \mathbf{2.2}$ is a generic general–sum strategic–form game Γ with two agents. Consider the following non–perturbed strategies (in the case strategies are perturbed, but not fully mixed, the proof is analogous): $\mathbf{x}_1$ prescribes that action L is played with a probability of one, and $\mathbf{x}_2$ prescribes that action l is played with a probability of one. For all the other actions, $\mathbf{x}_1, \mathbf{x}_2$ prescribe a probability of zero. Strategies $\mathbf{x}_1, \mathbf{x}_2$ constitute an SPE if and only if the subgame starting at $h = \mathbf{1.2}$ admits an NE that provides agent 2 an expected utility smaller than 1. Since $\mathbf{x}_1, \mathbf{x}_2$ prescribe a probability of zero in Γ, they do not pose any constrain over the problem to search for an NE for Γ providing agent 2 with no more than 1. Hence, our problem reduces to the problem to decide whether there is a NE with some property. □

The above theorem can be easily extended showing that the verification of an SE (with an arbitrary number of agents) is $\mathcal{NP}$–complete when the input is in sequence form. Furthermore, it is trivial to show that, when the input to the verification problem is a fully perturbed sequence form strategy profile, we have positive results. Indeed, given a fully mixed perturbed sequence form strategy, we can always derive an equivalent perturbed behavioral strategy and from this a non–perturbed behavioral strategy. Thus, we can apply the positive results with agent form.

Now, we consider the problem to verify a QPE. While this problem appears hard by working with the agent form, we have a tractable result with the sequence form.

THEOREM 4.2. *Given a game with two agents, it is in $\mathcal{P}$ the problem to decide whether or not a strategy profile $\boldsymbol{\sigma}$ is a QPE.*

Proof. In order to verify whether a strategy profile $\sigma = (\sigma_1, \sigma_2)$ is a QPE, we need to verify:

- the existence of a perturbed $\sigma_1(\epsilon)$ such that $\sigma_1(\epsilon) \to \sigma_1$ as $\epsilon \to 0$ and σ_2 is a best response to $\sigma_1(\epsilon)$,
- the existence of a perturbed $\sigma_2(\epsilon)$ such that $\sigma_2(\epsilon) \to \sigma_2$ as $\epsilon \to 0$ and σ_1 is a best response to $\sigma_2(\epsilon)$.

This is equivalent to verify the existence of a *lexicographic belief structure* according to [1, 13]. Since characterization of a QPE can be accomplished in sequence form without resorting the agent form we can formulate our problem with the sequence form exploiting the LCP formulation discussed in Appendix B. We can formulate the search for $\sigma_1(\epsilon)$ and $\sigma_2(\epsilon)$ as the search for two perturbed strategies $\mathbf{x}_1(\epsilon)$ and $\mathbf{x}_2(\epsilon)$ such that the following constraints hold (the constraints over $\mathbf{x}_2(\epsilon)$ are analogous):

$$F_1\mathbf{x}_1(\epsilon) = \mathbf{f}_1 \tag{12}$$

$$\mathbf{x}_1(\epsilon) >_L \mathbf{0} \tag{13}$$

$$F_2^T\mathbf{v}_2(\epsilon) - U_2^T\mathbf{x}_1(\epsilon) \geq_L \mathbf{0} \tag{14}$$

$$(F_2^T\mathbf{v}_2(\epsilon) - U_2^T\mathbf{x}_1(\epsilon))_q = 0 \qquad \forall q \in Q_2, \sigma_{2,a(q)} > 0 \tag{15}$$

$$\min_{x_{1,q}(\epsilon^k)>0} k < \min_{x_{1,q'}(\epsilon^k)>0} k \qquad \begin{array}{l} \forall a(q) \in \rho(w), a(q') \in \rho(w'), \\ w, w' \in H_{1,h}, h \in H_1, \\ \sigma_{1,a(q)} > 0, \sigma_{1,a(q')} = 0 \end{array} \tag{16}$$

where constraints (12) state that the strategy is well defined according to sequence form definition; constraints (13) state that the strategy is fully mixed ($>_L$ means 'lexico–positive'); constraints (14) are the dual best response constraints; constraints (15) state that, if the behavioral strategy $\sigma_{2,a(q)}$ has strictly positive value for the last action of sequence q, then the best response constraint associated with q must hold with equality; constraints (16) provide a hierarchical structure over the lexicographic perturbation of $\mathbf{x}_1(\epsilon)$ forcing in every information set that the minimum degree k, such that $x_{1,q}(\epsilon^k)$ is positive when the last action action $a(q)$ of q is played with $\sigma_{1,a(q)} > 0$, is strictly lower than the minimum degree k' related to sequences q's whose last action is played with $\sigma_{1,a(q')} = 0$.

The above feasibility problem can be solved iteratively as follows. At each iteration k, we find the values of $\mathbf{x}_1(\epsilon^k)$. Each iteration can be formulated as a linear mathematical programming problem. Iteration $k = 0$ requires the resolution of the following mathematical program:

$$F_1\mathbf{x}_1(\epsilon^0) = \mathbf{f}_1 \tag{17}$$

$$x_{1,q}(\epsilon^0) \geq 0 \qquad \forall a(q) \in A_1, \sigma_{1,a(q)} > 0 \tag{18}$$

$$x_{1,q}(\epsilon^0) = 0 \qquad \forall a(q) \in A_1, \sigma_{1,a(q)} = 0 \tag{19}$$

$$F_2^T\mathbf{v}_2(\epsilon^0) - U_2^T\mathbf{x}_1(\epsilon^0) \geq \mathbf{0} \tag{20}$$

$$(F_2^T\mathbf{v}_2(\epsilon^0) - U_2^T\mathbf{x}_1(\epsilon^0))_q = 0 \qquad \forall q \in Q_2, \sigma_{2,a(q)} > 0 \tag{21}$$

where constraints (17) are analogous to (12); constraints (18) and (19) correspond to (16); constraints (20) and (21) correspond to (14) and (15). The above program is feasible if $\boldsymbol{\sigma}$ is a Nash equilibrium. Therefore, if the above program is infeasible, then the algorithm stops and $\boldsymbol{\sigma}$ is not a QPE.

From $k = 1$ on, the mathematical program to solve is:

$$\max \sum_{\forall k'<k, x_{1,q}(\epsilon^{k'})=0} x_{1,q}(\epsilon^k) \tag{22}$$

$$F_1\mathbf{x}_1(\epsilon^k) = 0 \tag{23}$$

$$x_{1,q}(\epsilon^k) \geq 0 \qquad \forall q \in Q_1, x_{1,q}(\epsilon^{k'}) = 0, k' < k \tag{24}$$

$$x_{1,q}(\epsilon^k) \leq 1 \qquad \forall q \in Q_1 \tag{25}$$

$$x_{1,q}(\epsilon^k) = 0 \qquad \begin{array}{l} \forall q, q' \in Q_1, w, w' \in V_{1,h}, \\ a(q) \in \rho(w), a(q') \in \rho(w'), \\ \sigma_{1,a(q)} = 0, \sigma_{1,a(q')} > 0, \\ x_{1,q'}(\epsilon^{k'}) = 0, k' < k, h \in H_1 \end{array} \tag{26}$$

$$(F_2^T\mathbf{v}_2(\epsilon^k) - U_2^T\mathbf{x}_1(\epsilon^k))_q \geq 0 \qquad \begin{array}{l} \forall q \in Q_2, (F_2^T\mathbf{v}_2(\epsilon^{k'}) - \\ \qquad U_2^T\mathbf{x}_1(\epsilon^{k'}))_q = 0, k' < k \end{array} \tag{27}$$

$$(F_2^T\mathbf{v}_2(\epsilon^k) - U_2^T\mathbf{x}_1(\epsilon^k))_q = 0 \qquad \forall q \in Q_2, \sigma_{2,a(q)} > 0 \tag{28}$$

where constraints (23) grant the strategy to be well defined; constraints (24) grant that $\mathbf{x}_1(\epsilon)$ is lexico–positive; constraints (25) pose an upper bound of 1 over the coefficients of ϵ^k (this value does not affect the feasibility of the problem); constraints (26) force constraints (16); constraints (27) and (28) force constraints (14) and (15), respectively. The objective function aims at maximizing the sum of the coefficients $x_{1,q}(\epsilon^k)$ such that $x_{1,q}(\epsilon^{k'}) = 0$ for all $k' < k$.

The algorithm stops either when $\mathbf{x}_1(\epsilon)$ is strictly lexico–positive or when the objective function is 0. In the latter case, it is not possible to find any strictly lexico–positive $\mathbf{x}_1(\epsilon)$ that satisfies the above constraints and therefore $\boldsymbol{\sigma}$ is not a QPE. Otherwise, if there are strictly lexico–positive $\mathbf{x}_1(\epsilon)$ and $\mathbf{x}_2(\epsilon)$, $\boldsymbol{\sigma}$ is a QPE.

We discuss the completeness of the algorithm. Note that the constraints at iteration k depend on the solutions of the optimization problems at the previous iterations. Given that a linear optimization problem can admit different optimal solutions, we have that the possible paths the algorithm can follow are different. However, it can be observed that the set of constraints strictly relaxes from iteration k to k'. Therefore, for all the paths the algorithm can follow, the algorithm always terminates with the same outcome in terms of existence or non–existence of a strictly lexico–positive $\mathbf{x}_1(\epsilon)$ (notice that in the case of existence, different paths may lead to different strictly lexico–positive strategies).

Finally, we show that the number of iteration is in the worst case linear in the size of the game. At each iteration k, either some $x_{1,q}(\epsilon^k)$ that is zero for every $k' < k$ becomes strictly positive or the algorithm stops with failure.

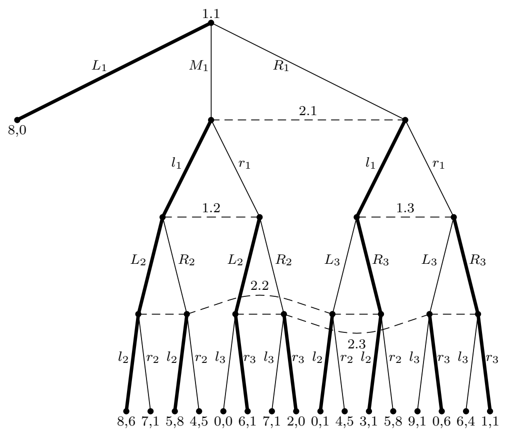


Figure 5: Example of strategy profile $\boldsymbol{\sigma}$ expressed in behavioral strategies that is a quasi perfect equilibrium ($\boldsymbol{\sigma}$ is represented by using bold lines to denote actions played with positive probability).

In the worst case, only one sequence becomes strictly lexico–positive per iteration and therefore the number of iteration is equal to the number of sequences. Thus, linear mathematical programming being polynomial time, the theorem is proved. □

We provide two examples to which we apply the algorithm described in the proof of Theorem 4.2.

Example 4.3. *Consider the game depicted in Fig. 3 and the strategy profile $\boldsymbol{\sigma} = (\sigma_{1,M_1} = 1, \sigma_{1,L_2} = 1, \sigma_{1,R_3} = 1, \sigma_{2,l_1} = 1, \sigma_{2,l_2} = 1, \sigma_{2,r_3} = 1)$. (As shown in Example 3.3, it is an SE.) We check whether or not it is a QPE. From the application of the algorithm we provide in the proof of Theorem 4.2, we obtain the following $\mathbf{x}_2(\epsilon)$:*

	l_1	r_1	l_2	r_2	l_3	r_3
ϵ^0	1	0	1	0	0	0
ϵ^1	0	0	0	0	0	0

.

At iteration 1, the algorithm stops because the objective function is zero. Indeed, the algorithm cannot put a positive value on r_2 without violating the constraints of best response of agent 1. As a result, no fully mixed $\sigma_2(\epsilon)$ makes σ_1 to be a best response and, therefore, $\boldsymbol{\sigma}$ is not a QPE.

Example 4.4. *Consider the game depicted in Fig. 5 and the strategy profile $\boldsymbol{\sigma} = (\sigma_{1,L_1} = 1, \sigma_{1,L_2} = 1, \sigma_{1,R_3} = 1, \sigma_{2,l_1} = 1, \sigma_{2,l_2} = 1, \sigma_{2,r_3} = 1)$. We check whether or not it is a QPE. From the application of the algorithm we provide in the proof of Theorem 4.2, we obtain the following fully mixed $\mathbf{x}_2(\epsilon)$:*

	l_1	r_1	l_2	r_2	l_3	r_3
ϵ^0	1	0	1	0	0	0
ϵ^1	−1	1	−2	1	0	1
ϵ^2	0	0	0	0	1	−1

,

and the following fully mixed $\mathbf{x}_1(\epsilon)$:

	L_1	M_1	R_1	L_2	R_2	L_3	R_3
ϵ^0	1	0	0	0	0	0	0
ϵ^1	−2	1	1	1	0	0	1
ϵ^2	0	0	0	−1	1	1	−1

,

therefore $\boldsymbol{\sigma}$ is a QPE.

Finally, we show that the employment of sequence form, when each agent takes into account also her own perturbations, presents several problems to verify an EFPE.

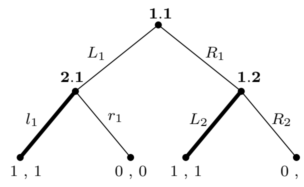


Figure 6: Game used in the proof of Proposition 4.5.

Proposition 4.5. *The best response optimization problem with the sequence form, when each agent takes into account also her own perturbations, cannot be used to verify an EFPE.*

Proof. Consider the game tree depicted in Fig. 6. At $h = \mathbf{1.2}$, the unique optimal strategy is $\sigma_{1,L_2} = 1$. Analogously, at $h = \mathbf{2.1}$, the unique optimal strategy is $\sigma_{2,l_1} = 1$. At $h = \mathbf{1.1}$, L_1 and/or R_1 can be optimal on the basis of the perturbation at the two subgames. For instance, with a perturbation over behavioral strategies such that $l_{2,r_1} = \epsilon^2$ and $l_{1,R_2} = \epsilon$, L_1 is strictly better than R_1 for agent 1. We show that maximizing over the expected utility provided by the sequences L_1 cannot be an optimal action. The expected utility, considering also the own perturbation, provided by sequence L_1 is: $EU_1(L_1) = (1 - l_{1,R_1}(\epsilon))(1 - l_{2,r_1}(\epsilon)) + l_{1,R_1}(\epsilon) - l_{1,R_2}(\epsilon) = 1 - l_{1,R_2}(\epsilon) - l_{2,r_1}(\epsilon) + l_{1,R_1}(\epsilon)l_{2,r_1}(\epsilon)$. The expected utility, considering also the own perturbation, provided by sequence R_1 is: $EU_1(R_1) = l_{1,L_1}(\epsilon)(1 - l_{2,r_1}(\epsilon)) + 1 - l_{1,L_1}(\epsilon) - l_{1,R_2}(\epsilon) = 1 - l_{1,R_2}(\epsilon) - l_{1,L_1}(\epsilon)l_{2,r_1}(\epsilon)$. It can be observed that for every possible combination of $l_{1,L_1}(\epsilon), l_{1,R_1}(\epsilon), l_{2,r_1}(\epsilon), l_{1,R_2}(\epsilon)$ the inequality $EU_1(R_1) > EU_1(L_1)$ holds, since $EU_1(R_1) - EU_1(L_1) = l_{2,r_1}(\epsilon)(1 - l_{1,L_1}(\epsilon) - l_{1,R_1}(\epsilon))$. Therefore, by maximizing over perturbed sequences we cannot verify correctly any EFPE that prescribes $\sigma_{1,L_1} = 1$. □

Notice that the above result does not show that the sequence form cannot be used to verify an EFPE at all, but that, if applicable, the sequence form must be rethought for this problem (and for the problem to compute an EFPE).

5. CONCLUSIONS AND FUTURE WORKS

We studied the problem to verify whether a solution is a given solution concept refining the NE for extensive–form games. This problem is of extraordinary importance. If the verification of a solution concept is intractable, such a solution concept cannot be adopted in practice. While verifying a NE is easy, this may be not the case for NE refinements. In this paper, we complete the results known in the literature concerning the verification of a SE and of an QPE, proving that problems to verify an SE with an arbitrary number of agents and a QPE with two agents are in $\mathcal{P}$ and we provide two pertinent algorithms based on linear programming. We show also that when the input solution is expressed in (non–perturbed) sequence form even verifying an SPE is $\mathcal{NP}$–complete and that sequence form, if applicable, must be rethought for the verification of an EFPE.

In future, we aim at completing our results, exploring the verification of an EFPE with two agents and of a Myerson's proper equilibrium with two agents [8].

6. REFERENCES

[1] L. Blum, A. Brandenburger, and E. Dekel. Lexicographic probabilities and equilibrium refinements. *ECONOMETRICA*, 59(1):81–98, 1991.

[2] S. Ceppi, N. Gatti, G. Patrini, and M. Rocco. Local search techniques for computing equilibria in two–player general–sum strategic–form games. In *AAMAS*, pages 1469–1470, 2010.

[3] R. Chandrasekaran. Total unimodularity of matrices. *SIAM J APPL MATH*, 17(6):1032–1034, 1969.

[4] X. Chen, X. Deng, and S. Teng. Computing Nash equilibria: approximation and smoothed complexity. In *FOCS*, pages 603–612, 2006.

[5] V. Conitzer and T. Sandholm. New complexity results about Nash equilibria. *GAME ECON BEHAV*, 63(2):621–641, 2008.

[6] C. Daskalakis, P. Goldberg, and C. Papadimitriou. The complexity of computing a Nash equilibrium. In *STOC*, pages 71–78, 2006.

[7] C. Daskalakis, A. Mehta, and C. Papadimitriou. A note on approximate Nash equilibria. *THEOR COMPUT SCI.*, 410(17):1581–1588, 2009.

[8] D. Fudenberg and J. Tirole. *Game Theory.* 1991.

[9] N. Gatti and C. Iuliano. Computing an extensive–form perfect equilibrium in two–player games. In *AAAI*, 2011.

[10] A. Ghouila-Houri. Caracterisation des matrices totalement unimodulaires. *C. R. Acad. Sci. Paris*, 254:1192–1194, 1962.

[11] A. Gilpin and T. Sandholm. Lossless abstraction of imperfect information games. *J ACM*, 54(5):25, 2007.

[12] A. Gilpin, T. Sandholm, and T. Sørensen. Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of texas hold'em poker. In *AAAI*, pages 50–57, 2007.

[13] S. Govindan and T. Klumpp. Perfect equilibrium and lexicographic beliefs. *INT J GAME THEORY*, 31(2):229–243, 2003.

[14] K. Hansen, P. Miltersen, and T. Sørensen. The computational complexity of trembling hand perfection and other equilibrium refinments. In *SAGT*, pages 198–209, 2010.

[15] E. Kohlberg and P. Reny. Independence on relative probability spaces and consistent assessments in game trees. *J ECON THEORY*, 75(2):280–313, 1997.

[16] D. R. Kreps and R. Wilson. Sequential equilibria. *ECONOMETRICA*, 50(4):863–894, 1982.

[17] H. Kuhn. Extensive games. *Prooceedings of the National Academy of Sciences*, 36:570–576, 1950.

[18] C. Lemke. Some pivot schemes for the linear complementarity problem. *Mathematical Programming Study*, 7:15–35, 1978.

[19] C. Lemke and J. Howson. Equilibrium points of bimatrix games. *SIAM J APPL MATH*, 12(2):413–423, 1964.

[20] P. Miltersen and T. Sørensen. Computing a quasi–perfect equilibrium of a two–player game. *ECON THEOR*, 42(1):175–192, 2010.

[21] R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a Nash equilibrium. In *AAAI*, pages 664–669, 2004.

[22] T. Sandholm, A. Gilpin, and V. Conitzer. Mixed–integer programming methods for finding Nash equilibria. In *AAAI*, pages 495–501, 2005.

[23] R. Selten. Reexamination of the perfectness concept for equilibrium points in extensive games. *INT J GAME THEORY*, 4(1):25–55, 1975.

[24] E. van Damme. A relation between perfect equilibria in extensive form games and proper equilibria in normal form games. *INT J GAME THEORY*, 13(2):1—13, 1984.

[25] E. van Damme. *Stability and Perfection of Nash Equlibria.* Springer, 1991.

[26] J. von Neumann and O. Morgenstern. *Theory of games and economic behavior.* Princeton University Press, 1944.

[27] B. von Stengel. Efficient computation of behavior strategies. *INT J GAME THEORY*, 14(2):220–246, 1996.

[28] B. von Stengel. *Algorithmic Game Theory*, chapter Equilibrium computation for two–player games in strategic and extensive form, pages 53–78. Cambridge, 2007.

[29] L. Wolsey. *Integer Programming.* Wiley–Interscience, 1998.

APPENDIX

A. LEXICOGRAPHIC PERTURBATIONS

Given an ordered vector $\mathbf{z}_1 \in \mathbb{R}^n$, we say that $\mathbf{z}_1$ is *lexico–positive* if the first non–zero element of $\mathbf{z}_1$ is positive. Formally, we write $\mathbf{z}_1 \geq_L \mathbf{0}$. $\mathbf{z}_1$ is strictly lexico–positive if it is lexico–positive and there is at least a strictly positive element. Easily, given a pair of ordered vectors $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^n$, we say that $\mathbf{z}_1 \geq_L \mathbf{z}_2$ if and only if $\mathbf{z}_1 - \mathbf{z}_2$ is lexico positive.

A perturbation $l_{i,a}(\epsilon)$ over action a is a polynomial in ϵ, e.g., $l_{i,a}(\epsilon) = c_1\epsilon + c_2\epsilon^2 + c_3\epsilon^3 + c_4\epsilon^4 + \ldots$, where $c_k \in \mathbb{R}$. A perturbation can be represented as one ordered vector in which the first element is the coefficient c_1 of ϵ, the second element is the coefficient c_2 of ϵ^2, and so on. That is, when ϵ goes to zero, $c_{k_1}\epsilon^{k_1}$ and $c_{k_2}\epsilon^{k_2}$ are comparable if and only if $k_1 = k_2$. Similarly, a perturbed strategy $\boldsymbol{\sigma}_i(\epsilon)$ (analogously, $\mathbf{x}_i(\epsilon)$) can be represented by using an lexico positive ordered vector per action (sequence). Requiring that a perturbed strategy $\boldsymbol{\sigma}_i(\epsilon)$ (analogously, $\mathbf{x}_i(\epsilon)$) is fully mixed is equivalent to requiring that each element $\sigma_{i,a}(\epsilon)$ is strictly lexico positive.

B. QPE COMPUTATION

A QPE with two–agent games can be computed by applying a specific symbolic perturbation $\mathbf{l}_1(\epsilon), \mathbf{l}_2(\epsilon)$ (see [20] for the details on the perturbation) to the LCP to find an NE (the solving algorithm is the same for the computation of NE). Given a perturbed strategy $\mathbf{x}_i \geq \mathbf{l}_i(\epsilon)$, we substitute $\mathbf{x}_i$ with $\tilde{\mathbf{x}}_i + \mathbf{l}_i(\epsilon)$ where $\tilde{\mathbf{x}}_i \geq \mathbf{0}$. The resulting symbolically perturbed LCP is:

$$\tilde{\mathbf{x}}_i \geq \mathbf{0} \qquad \forall i \in \{1,2\} \qquad (29)$$

$$F_i \tilde{\mathbf{x}}_i = \mathbf{f}_i - F_i \mathbf{l}_i(\epsilon) \qquad \forall i \in \{1,2\} \qquad (30)$$

$$F_i^T \mathbf{v}_i - U_i \tilde{\mathbf{x}}_{-i} \geq \mathbf{0} + U_i \mathbf{l}_{-i}(\epsilon) \qquad \forall i \in \{1,2\} \qquad (31)$$

$$\tilde{\mathbf{x}}_i^T \cdot (F_i^T \mathbf{v}_i - U_i \tilde{\mathbf{x}}_{-i} - U_i \mathbf{l}_{-i}(\epsilon)) = 0 \qquad \forall i \in \{1,2\} \qquad (32)$$

where $\mathbf{v}_i$ are the dual variables of the best response optimization problems and their values are the expected utilities associated with the best actions for each information set of agent i.

Playing Repeated Stackelberg Games with Unknown Opponents

Janusz Marecki
IBM T.J.Watson Research
P.O. Box 218
Yorktown Heights, NY 10598
marecki@us.ibm.com

Gerry Tesauro
IBM T.J.Watson Research
P.O. Box 218
Yorktown Heights, NY 10598
gtesauro@us.ibm.com

Richard Segal
IBM T.J.Watson Research
P.O. Box 218
Yorktown Heights, NY 10598
rsegal@us.ibm.com

ABSTRACT

In Stackelberg games, a "leader" player first chooses a mixed strategy to commit to, then a "follower" player responds based on the observed leader strategy. Notable strides have been made in scaling up the algorithms for such games, but the problem of finding optimal leader strategies spanning multiple rounds of the game, with a Bayesian prior over unknown follower preferences, has been left unaddressed. Towards remedying this shortcoming we propose a first-of-a-kind tractable method to compute an optimal plan of leader actions in a repeated game against an unknown follower, assuming that the follower plays myopic best-response in every round. Our approach combines Monte Carlo Tree Search, dealing with leader exploration/exploitation tradeoffs, with a novel technique for the identification and pruning of dominated leader strategies. The method provably finds asymptotically optimal solutions and scales up to real world security games spanning double-digit number of rounds.

Categories and Subject Descriptors

G [**3**]: Probabilistic algorithms (including Monte Carlo)

General Terms

Algorithms

Keywords

Stackelberg Games, Monte-Carlo Tree Search

1. INTRODUCTION

Recent years have seen a rise in interest in applying game theoretic models to real world security domains, ranging from allocation of security checkpoints at Los Angeles International Airport [13] to the analysis and detection of computer network intrusions [1, 10]. As these security domains impose non-simultaneous player actions, they are naturally modeled as Stackelberg games [4] wherein one player (referred to as the leader) commits to a mixed strategy of its choice, while the second player (referred to as the follower) responds based on the observed leader strategy. This type of approach has received a lot of attention, resulting in new methods for the scale-up of the proposed algorithms to games with large number of follower types [12] or large leader strategy spaces [6].

In arriving at the optimal leader strategies for such games, of critical importance is the leader's ability to profile the follower [14]. In essence, determining the preferences of the follower actions is a necessary step in predicting the follower best responses to leader actions, which in turn are necessary for finding the optimal leader strategy. If these follower preferences cannot be determined exactly, one can consider Stackelberg formulations with distributional uncertainty over the follower payoffs [14]. However, the best leader strategies in such games are often conservative [7] and prior work only considered the case of single-round games. The repeated-game Stackelberg scenario, wherein the leader can exploit extra information in the form of the follower responses, was not studied. A notable advance for repeated games was recently reported in [9], wherein the authors develop an elegant method for choosing leader strategies to uncover the follower preferences in as few rounds as possible. To our knowledge, however, no attempts have been made to *exploit* the revealed information about follower preferences to optimize total leader payoff over the rounds of the game.

This paper remedies these shortcomings by providing a first-of-a-kind method to balance the exploration of the follower payoff structure versus the exploitation of this knowledge to optimize on-the-fly the expected cumulative reward-to-go of the leader. By coupling Monte-Carlo Tree Search sampling to estimate the utility of leader mixed strategies with preemptive pruning of dominated leader strategies, we show how to effectively handle a broad class of repeated Stackelberg games often employed to model real world domains. We first provide a brief formal description of the decision problems at hand and recall the Bayesian Stackelberg game model. We then develop our algorithm and a separate method for pruning of dominated leader strategies. We finally provide an empirical evaluation of our method and discuss our results in the context of related work.

2. PROBLEM STATEMENT

2.1 Bayesian Stackelberg Games

A Bayesian Stackelberg game assumes a leader agent of a single type and a follower agent of type drawn from a set Θ. The set of pure strategies of the leader is $A_l = \{a_{l_1}, ..., a_{l_M}\}$ and the set of pure strategies of the follower is $A_f = \{a_{f_1}, ..., a_{f_N}\}$. Game payoffs are described in terms of the player utility functions: Leader's utility function is

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

$u_l : A_l \times A_f \to \mathbb{R}$ while the follower utility function $u_f : \Theta \times A_l \times A_f \to \mathbb{R}$ is unique for each type $\theta \in \Theta$ of the follower. The leader acts first by committing to a mixed strategy $\sigma \in \Sigma$ where $\sigma(a_l)$ is the probability of the leader executing its pure strategy $a_l \in A_l$. (Mixed strategies allow for higher expected payoffs of the leader as shown in [12].) For a given leader strategy σ and a follower of type $\theta \in \Theta$, the follower's best response $B(\theta, \sigma) \in A_f$ to σ is a pure strategy $B(\theta, \sigma) \in A_f$ that satisfies:

$$B(\theta, \sigma) = \arg \max_{a_f \in A_f} \sum_{a_l \in A_l} \sigma(a_l) u_f(\theta, a_l, a_f). \tag{1}$$

Given the follower type $\theta \in \Theta$, the expected utility of the leader strategy σ is therefore given by:

$$U(\theta, \sigma) = \sum_{a_l \in A_l} \sigma(a_l) u_l(a_l, B(\theta, \sigma)). \tag{2}$$

Given a probability distribution $P(\theta)$ over the follower types, the expected utility of the leader strategy σ over all the follower types is hence:

$$U(\sigma) = \sum_{\theta \in \Theta} P(\theta) \sum_{a_l \in A_l} \sigma(a_l) u_l(a_l, B(\theta, \sigma)). \tag{3}$$

Solving a single-round Bayesian Stackelberg game involves finding $\sigma^* = \arg\max_{\sigma \in \Sigma} U(\sigma)$.

An example Stackelberg game is depicted in Figure 1. Here, the leader agent (the security force) first commits to a mixed strategy. The follower agent (the adversary) of just a single type then observes the leader strategy and responds optimally to it, with a pure strategy, to maximize its own payoff. For example, the leader mixed strategy to "Patrol Terminal #1" (abbr. "PT1") with probability 0.5 and "Patrol Terminal #2" with probability 0.5 provokes the follower response "Attack Terminal #1" (abbr. "AT1"), because it provides the follower with the expected utility of $0.5 \cdot (-2) + 0.5 \cdot (2) = 0$ which is greater than the expected utility of $0.5 \cdot (2) + 0.5 \cdot (-4) = -1$ if the follower were to "Attack Terminal #2". One can calculate that the best leader mixed strategy for the Stackelberg game in Figure 1 is a pair [PT1=60%, PT2=40%] which (assuming that the follower breaks ties in the leader's favor) provokes the follower response AT1 thus providing the leader with the expected utility of $EU([PT1 = 60\%, PT2 = 40\%]) = 0.6 * 6 + 0.4 * 3 = 4.8$. (Note, how this mixed strategy is superior to the leader pure strategies [PT1=100%, PT2=0%] and [PT1= 0%, PT2=100%] which illustrates the benefits of randomized strategies in security domains.)

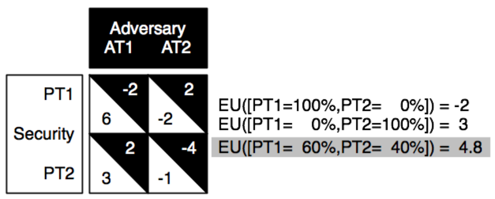

Adversary
AT1
AT2
PT1
Security
PT2
-2
2
6
-2
2
-4
3
-1
EU([PT1=100%,PT2= 0%]) = -2
EU([PT1= 0%,PT2=100%]) = 3
EU([PT1= 60%,PT2= 40%]) = 4.8

Figure 1: Single round Stackelberg game

2.2 Repeated Game Formulation

We adopt the model of repeated Bayesian Stackelberg games [9] which assumes that nature draws a follower of type $\theta \in \Theta$ at the start of the game, and then the leader plays H rounds of a Stackelberg game against a follower with fixed type θ. The formulation also posits that the follower plays a myopic (non-strategic) best-response strategy to the leader strategy observed in each round. (We defer to future work the more general case where the follower may also behave strategically, and may utilize a distribution over unknown leader preferences.) As such, the leader may never know the actual follower type θ that it is playing against, but it can infer the parts of the opponent payoff structure by observing follower responses to various leader actions. Whereas the objective in [9] was to minimize the number of rounds needed to exactly identify the follower type θ, our objective is to compute the best leader strategy in each round, given $P(\theta)$ and history of play in prior rounds, to maximize total leader payoff over H rounds of play.

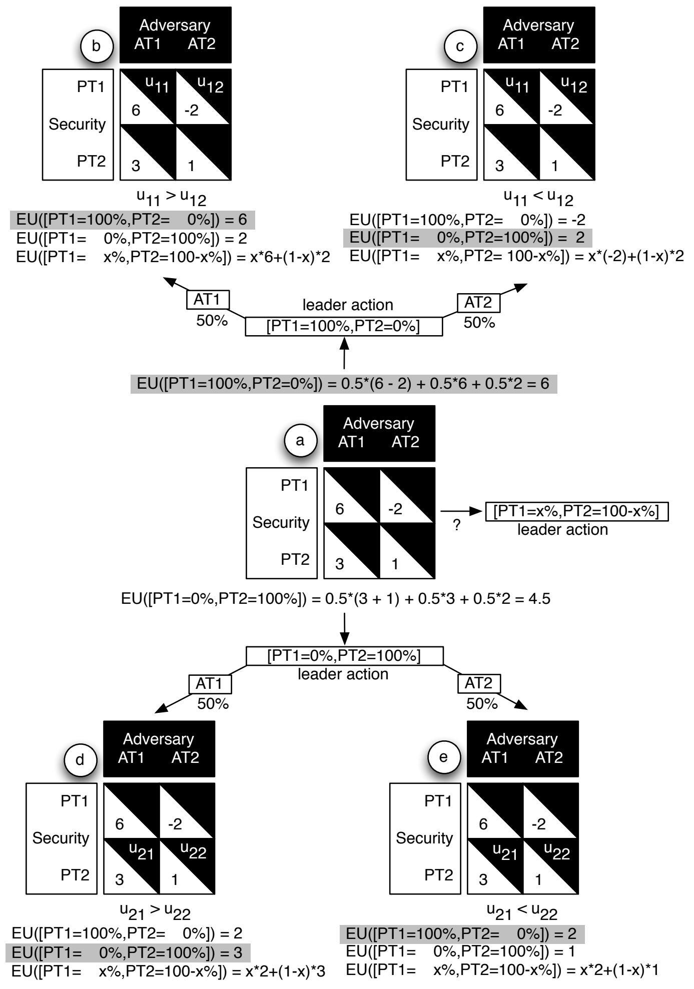
To illustrate this concept refer to a two-round Stackelberg game in Figure 2 where the follower payoffs are initially unknown (uniformly distributed), represented by missing values in Table (a). Suppose the leader action in the first round is [PT1=100%,PT2=0%]. If the follower responds with action AT1 (refer to Table (b)), the leader receives a first-round payoff of 6, and infers that $u_{11} > u_{12}$. Consequently, in the second round the leader will again play [PT1=100%,PT2=0%], for it assuredly provokes the follower response AT1 and hence provides the leader with another payoff of 6, for a total two-round payoff of 12. (Note, that as the leader strategy [PT1=100%,PT2=0%] will not provide additional information about the follower payoffs, it will not be chosen in the second round of the game by the algorithm introduced in [9].) On the other hand, if the follower replies in the first round to [PT1=100%,PT2=0%] by playing AT2 (refer to Table (c)), the leader receives first-round payoff of -2, and infers that $u_{11} < u_{12}$. In that case, the optimal leader strategy in the *second* round is [PT1=0%,PT2=100%] yielding an expected payoff of $0.5 \cdot (3 + 1) = 2$ in the second round, for a two-round total of 0. We conclude that, after initially playing [PT1=100%,PT2=0%], the leader has 50% chance each of receiving total payoff of either 12 or 0, so the leader's expected total payoff is 6. Similarly, one can derive that the expected total utility of the leader strategy [PT1=0%,PT2=100%] in the first round of the game (refer to Tables (d) and (e)) is EU([PT1=0%,PT2=100%])=4.5.

Unfortunately, finding the optimal leader strategy in the first round of the game requires one to evaluate all the strategies [PT1=x,PT2=1-x], $0 \leq x \leq 1$, considering for each strategy the unique knowledge of the opponent payoff structure gained by observing the follower responses to the said strategies. As the derivation of the formulae for the the expected utilities of the leader actions for arbitrary repeated Stackelberg games is an open research problem, we propose to approach the problem using a customized version of the Monte-Carlo Tree Search method, as shown next.

3. MCTS APPROACH

3.1 MCTS Overview

Monte-Carlo Tree Search (MCTS) methods provide new tools for online planning in complex sequential decision problems that have generated considerable excitement in recent years, due to breakthrough results in challenging domains such as 19 × 19 Go [5] and General Game Playing [3]. The key innovation of MCTS is to incorporate node evaluations within traditional tree search techniques that are based on



b
c
a
d
e
Adversary
AT1 AT2
PT1
Security
PT2
$u_{11} > u_{12}$
EU([PT1=100%,PT2= 0%]) = 6
EU([PT1= 0%,PT2=100%]) = 2
EU([PT1= x%,PT2=100-x%]) = x*6+(1-x)*2
$u_{11} < u_{12}$
EU([PT1=100%,PT2= 0%]) = -2
EU([PT1= 0%,PT2=100%]) = 2
EU([PT1= x%,PT2= 100-x%]) = x*(-2)+(1-x)*2
leader action
[PT1=100%,PT2=0%]
AT1 50%
AT2 50%
EU([PT1=100%,PT2=0%]) = 0.5*(6 - 2) + 0.5*6 + 0.5*2 = 6
[PT1=x%,PT2=100-x%]
leader action
EU([PT1=0%,PT2=100%]) = 0.5*(3 + 1) + 0.5*3 + 0.5*2 = 4.5
[PT1=0%,PT2=100%]
$u_{21} > u_{22}$
EU([PT1=100%,PT2= 0%]) = 2
EU([PT1= 0%,PT2=100%]) = 3
EU([PT1= x%,PT2=100-x%]) = x*2+(1-x)*3
$u_{21} < u_{22}$
EU([PT1=100%,PT2= 0%]) = 2
EU([PT1= 0%,PT2=100%]) = 1
EU([PT1= x%,PT2=100-x%]) = x*2+(1-x)*1

Figure 2: Two-round Stackelberg game, with uniformly distributed follower payoffs. First round of play shown in (a). If the leader plays 100% PT1, the follower may reply either AT1, leading to (b), or AT2, leading to (c). If the leader plays 100% PT2, follower may reply either AT1, leading to (d), or AT2, leading to (e).

stochastic simulations (i.e., "rollouts" or "playouts"), while also using bandit-sampling algorithms to focus the bulk of simulations on the most promising branches of the tree search. This combination appears to have overcome traditional exponential scaling limits to established planning techniques in a number of large-scale domains. MCTS is also an anytime algorithm and simple parallelization schemes have been found to scale effectively to hundreds of cores [16].

Standard implementations of MCTS maintain and incrementally grow a collection of nodes, usually organized in a tree structure, representing possible states that could be encountered in the given domain. The nodes maintain counts n_{sa} of the number of simulated trials in which action a was selected in state s, as well as mean reward statistics $\bar{r}_{sa}$ obtained in those trials. A simulation trial begins at the root node, representing the current state, and steps of the trial descend the tree using a tree-search policy that is based on sampling algorithms for multi-armed bandits that embody a tradeoff between exploiting actions with high mean reward, and exploring actions with low sample counts. When the trial reaches the frontier of the tree, it may continue performing simulation steps by switching to a "playout policy," which commonly select actions using a combination of heuristics. When the trial terminates, sample counts and mean reward values are updated in all tree nodes that participated in the trial. At the end of all simulations, the reward-maximizing top-level action from the root of the tree is selected and performed in the real domain.

Our implementation of MCTS makes use of the UCT algorithm [8], which employs a tree-search policy based on a variant of the UCB1 bandit-sampling algorithm [2]. The policy computes an upper confidence bound B_{sa} for each possible action a in a given state s according to: $B_{sa} = \bar{r}_{sa} + c\sqrt{\ln N_s / n_{sa}}$, where $N_s = \sum_{a'} n_{sa'}$ is the total number of trials of all actions in the given state, and c is a tunable constant controlling the tradeoff between exploration and exploitation. With an appropriate choice of the value of c, UCT is guaranteed to converge to selecting the best top-level action with probability 1.

3.2 MCTS in Repeated Stackelberg Games

We now present our MCTS-based method for planning leader actions in repeated Stackelberg games with unknown opponents. A key feature of our method builds upon the assumption that the leader has a prior probability distribution over possible follower types (equivalently, over follower utility functions). We leverage this by performing MCTS trials in which each trial simulates the behavior of the follower using an independent draw from this distribution. As different follower types transition down different branches of the MCTS tree, this provides a simple and elegant means of implicitly approximating the posterior distribution for any given history in the tree, where the most accurate posteriors are focused on the most critical paths for optimal planning. This may enable faster approximately optimal planning than established methods which require fully specified transition models for all possible histories as input to the method.

A high-level depiction of the method is given in Figure 3. The method performs a total of T simulated trials, each with a randomly drawn follower, where a trial consists of H rounds of play. In each round, the leader chooses a mixed strategy $\sigma \in \Sigma$ to be performed, that is, to play each pure strategy $a_l \in A_l$ with probability $\sigma(a_l)$. To obtain a finite enumeration of leader mixed strategies, similarly to [11], we discretize the $\sigma(a_l)$ values into integer multiples of a discretization interval $\epsilon = 1/K$, and represent the leader mixed strategy components as $\sigma(a_l) = k_l \cdot \epsilon$ where $\{k_l\}$ is a set of non-negative integers s.t. $\sum k_l = K$. In the example in Figure 3 the number of leader *pure* strategies is $|A_l| = 2$ and $K = 2$ and the leader can choose to perform only one of the following three *mixed* strategies: $LA1 = [0.0, 1.0]$; $LA2 = [0.5, 0.5]$ or $LA3 = [1.0, 0.0]$ as shown in Figure 3. Upon observing the leader mixed strategy, the follower then plays a greedy pure-strategy response, that is, it selects from among its pure strategies (FR1, FR2, FR3 in Figure 3) the strategy achieving highest expected payoff given the observed leader mixed strategy. Although such discretization method can in theory lead to suboptimal solutions [11], the underlying discretization error is rarely seen in practice [12], especially for big values of K. An argument can also be made that in real world Stackelberg games the leader can implement its mixed strategy (and the follower can observe it) with only a limited precision [14].

Leader strategies in each round of each trial are selected by MCTS using either the UCB1 tree-search policy for the initial rounds within the tree, or a playout policy for the remaining rounds taking place outside the tree. Our playout policy uses uniform random selection of leader mixed strategies for each remaining round of the playout. We grow the MCTS tree incrementally with each trial, starting from just the root node at the first trial. Whenever a new leader mixed strategy is tried from a given node, the set of all possible transition nodes (i.e. leader mixed strategy followed by all possible follower responses) are added to the tree.

The basic idea of the abbreviated proof of convergence of our algorithm is to map a repeated Bayesian Stackelberg game to an equivalent Markov Decision Process. The MDP states in such a mapping are the finite horizon histories of pairs of (discretized) leader mixed strategies and their corresponding follower responses. The MDP actions represent possible leader mixed strategies in a current state. The transition probability from state $s = (h)$ to state $s' = (h|\sigma a_f)$ given action $\sigma \in \Sigma$ equals the probability of a follower response a_f to σ which can be uniquely determined from $P(\theta)$ and the observed history h. Finally, the corresponding MDP reward is given in Equation 2 wherein $B(\theta, \sigma)$ from Equation 1 is known to be a_f. Following Lemma 1 in [17], the expected payoff of a repeated Bayesian Stackelberg game policy equals the expected payoff of the equivalent MDP policy and therefore the optimal repeated Bayesian Stackelberg game policy and the optimal MDP policy are equivalent. Since our method samples according to the UCT formula, which is guaranteed to converge to the optimal MDP policy [8], therefore it also converges to the optimal repeated Bayesian Stackelberg game policy.

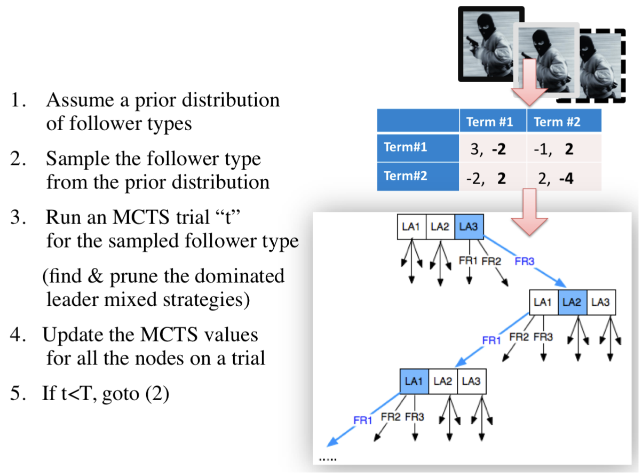

1. Assume a prior distribution of follower types
2. Sample the follower type from the prior distribution
3. Run an MCTS trial "t" for the sampled follower type
(find & prune the dominated leader mixed strategies)
4. Update the MCTS values for all the nodes on a trial
5. If t<T, goto (2)
Term #1
Term #2
Term#1
Term#2
3, -2
-1, 2
-2, 2
2, -4
LA1
LA2
LA3
FR1
FR2
FR3
.....

Figure 3: MCTS algorithm overview

4. PRUNING OF THE DOMINATED LEADER STRATEGIES

As it is shown in this section, in some cases, the leader's exploration of the complete reward structure of the follower is unnecessary. In essence, in any round of the game, the leader can identify the leader strategies—that have not yet been employed by the leader—whose immediate expected value for the leader is guaranteed not to exceed the expected value of leader strategies employed by the leader in the earlier rounds of the game. If the leader then just wants to maximize the expected payoff of its next action, these not-yet-employed strategies can safely be disregarded.

To formalize the concept of pruning of dominated leader strategies assume that the leader is playing a repeated Stackelberg game with a follower of type $\theta \in \Theta$. Furthermore, denote by $E^{(n)} \subset \Sigma$ a set of leader mixed strategies that have been employed by the leader in rounds $1, 2, ..., n$ of the game. Notice, that the leader who aims to maximize its payoff in the $n+1st$ round of the game should consider to employ an unused strategy $\sigma \in \Sigma - E^{(n)}$ only if:

$$\overline{U}(\theta, \sigma) > \max_{\sigma' \in E^{(n)}} U(\theta, \sigma') \qquad (4)$$

Where $\overline{U}(\theta, \sigma)$ is the upper bound on the expected utility of the leader playing σ, established from the leader observations $B(\theta, \sigma')$; $\sigma' \in E^{(n)}$ as follows:

$$\overline{U}(\theta, \sigma) = \max_{a_f \in A_f(\sigma)} U(\sigma, a_f). \qquad (5)$$

Where $A_f(\sigma) \subset A_f$ is defined here as a set of follower actions a_f that can still (given $B(\theta, \sigma')$; $\sigma' \in E^{(n)}$) constitute the follower best response to σ while $U(\sigma, a_f)$ is the expected utility of the leader mixed strategy σ if the follower responds to it by executing action a_f. That is:

$$U(\sigma, a_f) = \sum_{a_l \in A_l} \sigma(a_l) u_l(a_l, a_f) \qquad (6)$$

Thus, in order to determine whether a not-yet-employed strategy σ should be executed, one has to determine the elements of a best response set $A_f(\sigma)$ given $B(\theta, \sigma')$ for all $\sigma' \in E^{(n)}$. We now show how that can be accomplished.

4.1 Best Response Sets

To find the actions that can still constitute the best response of the follower of type θ to a given leader strategy σ, we first define the concept of *Best Response Sets* and *Best Response Anti-Sets* and then prove an important property of best response sets.

DEFINITION 1. *For each action $a_f \in A_f$ of the follower, a best response set Σ_{a_f} is a set of all the leader strategies $\sigma \in \Sigma$ for which it holds that $B(\theta, \sigma) = a_f$.*

DEFINITION 2. *For each action $a_f \in A_f$ of the follower, a best response anti-set $\overline{\Sigma}_{a_f}$ is a set of all the leader strategies $\sigma \in \Sigma$ for which it holds that $B(\theta, \sigma) \neq a_f$.*

PROPOSITION 1. *Each best response set Σ_{a_f} is convex and $\{\Sigma_{a_f}\}_{a_f \in A_f}$ is a finite covering of Σ.*

PROOF. By contradiction: If Σ_{a_f} is not convex then there must exist $\sigma', \sigma'' \in \Sigma_{a_f}$ such that $B(\theta, \sigma') = B(\theta, \sigma'') = a_f$ and $\sigma = \lambda\sigma' + (1-\lambda)\sigma''$; $\lambda > 0$ such that $\sigma \notin \Sigma_{a_f}$. From Equation (1) it then holds that:

$$\begin{aligned}
\sum_{a_l \in A_l} \sigma'(a_l) u_f(a_l, a_f) &> \sum_{a_l \in A_l} \sigma'(a_l) u_f(a_l, \overline{a}_f) \\
\sum_{a_l \in A_l} \sigma''(a_l) u_f(a_l, a_f) &> \sum_{a_l \in A_l} \sigma''(a_l) u_f(a_l, \overline{a}_f) \\
\sum_{a_l \in A_l} \sigma(a_l) u_f(a_l, \overline{a}_f) &> \sum_{a_l \in A_l} \sigma(a_l) u_f(a_l, a_f).
\end{aligned}$$

Where $\overline{a}_f \in A_f$ is not a_f. After adding these inequalities and substituting $\sigma := \lambda\sigma' + (1-\lambda)\sigma''$ we obtain:

$$\sum_{a_l \in A_l} \sigma(a_l) u_f(a_l, \overline{a}_f) \quad < \quad \sum_{a_l \in A_l} \sigma(a_l) u_f(a_l, a_f)$$

Which contradicts the earlier inequality. Now, since for each $\sigma \in \Sigma$ there exists some $a_f \in A_f$ such that $B(\theta, \sigma) = a_f$, we have that $\{\Sigma_{a_f}\}_{a_f \in A_f}$ covers the entire set Σ and is therefore a partitioning of Σ. $\square$

We first illustrate how to find the follower best responses on an example and then provide a method that achieves it in a general case. Specifically, we now illustrate that (after a few rounds of the games) there may indeed exist $\sigma \in \Sigma$ such that $A_f(\sigma) \neq A_f$. Consider an example in Figure 4 where the game has already been played for two rounds. Let $A_l = \{a_{l_1}, a_{l_2}\}$, $A_f = \{a_{f_1}, a_{f_2}, a_{f_3}\}$ and $E^{(2)} = \{\sigma', \sigma''\}$ where $\sigma'(a_{l_1}) = 0.25; \sigma'(a_{l_2}) = 0.75$ and $\sigma''(a_{l_1}) = 0.75; \sigma''(a_{l_2}) = 0.25$. Furthermore, assume $U(a_{l_1}, a_{f_1}) = 0$; $U(a_{l_2}, a_{f_1}) = 1$; $U(a_{l_1}, a_{f_2}) = 1$; $U(a_{l_2}, a_{f_2}) = 0$ and $U(a_{l_1}, a_{f_3}) = U(a_{l_2}, a_{f_3}) = 0$. The follower best responses observed so far are $B(\theta, \sigma') = a_{f_1}$ (solid black circle) and $B(\theta, \sigma'') = a_{f_2}$ (black circle with dashed perimeter).

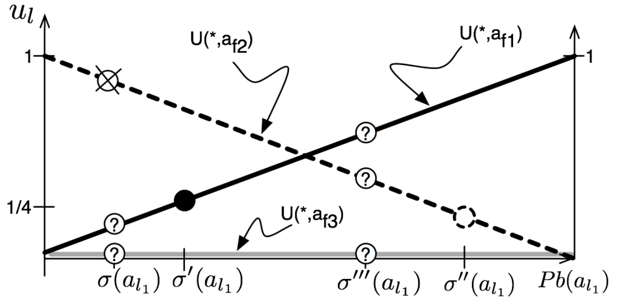


Figure 4: Best response actions

Notice, how in this context it is not profitable for the leader to employ a mixed strategy σ such that $\sigma(a_{l_1}) \in [0, \sigma'(a_{l_1})) \cup (\sigma''(a_{l_1}), 1]$. In particular, for σ such that $\sigma(a_{l_1}) \in [0, \sigma'(a_{l_1}))$ (refer to Figure 4 point σ) it holds that $B(\theta, \sigma) \neq a_{f_2}$ because otherwise (from Proposition (1)) the convex set $\Sigma_{a_{f_2}}$ would contain the elements σ and σ''—and hence also contain the element σ'—which is not true as $B(\theta, \sigma') = a_{f_1} \neq a_{f_2}$. Consequently, we have $A_f(\sigma) = \{a_{f_1}, a_{f_3}\}$ (notice the points with question marks above σ in Figure 4) which implies that $\overline{U}(\theta, \sigma) = \max\{U(\sigma, a_{f_1}), U(\sigma, a_{f_3})\} < \max\{0.25, 0\} = 0.25 = \max\{U(\sigma', a_{f_1}), U(\sigma'', a_{f_2})\}$. Hence, while employing strategy σ would allow the leader to learn $B(\theta, \sigma)$ (i.e., to disambiguate in Figure 4 the question marks in points above σ), this knowledge would not translate into the leader higher payoffs: The immediate expected reward for the leader for employing strategies σ', σ'' is always greater than the expected reward for employing σ such that $\sigma(a_{l_1}) \in [0, \sigma'(a_{l_1})) \cup (\sigma''(a_{l_1}), 1]$.

The example in Figure 4 also illustrates how the leader has to balance the benefits of exploration versus exploitation in the current round of the game. Specifically, the leader has a choice to either play one of the strategies σ', σ'' it had employed in the past (σ' if $U(\sigma', a_{f_1}) > U(\sigma'', a_{f_2})$ or σ'' otherwise) or play some strategy σ''' such that $\sigma'''(a_{l_1}) \in (\sigma'(a_{l_1}), \sigma''(a_{l_1})) = [0,1] \setminus [0, \sigma'(a_{l_1})) \setminus (\sigma''(a_{l_1}), 1]$ that it had not yet employed—and hence does not know what the follower best response $B(\theta, \sigma''')$ for this strategy is. Notice, that in this case, $A_f(\sigma''') = \{a_{f_1}, a_{f_2}, a_{f_3}\}$ (illustrated in Figure 4 by three points with question marks above σ'''). Now, if $B(\theta, \sigma''') = a_{f_3}$ were true, it would mean that $U(\sigma''', a_{f_3}) < \max\{U(\sigma', a_{f_1}), U(\sigma'', a_{f_2})\}$. In such case, the leader would explore the follower payoff preference (by learning $B(\theta, \sigma''')$) at a cost of loosing the potential immediate payoff of $U(\sigma''', a_{f_3}) - \max\{U(\sigma', a_{f_1}), U(\sigma'', a_{f_2})\}$.

Finally, the example also shows that although the immediate expected utility for executing a not-yet-employed strategy is smaller than the immediate expected utility for executing a strategy employed in the past, in some cases it might be profitable *not* to prune such not-yet-employed strategy. For example, if the game in Figure 4 is going to be played for at least two more rounds, the leader might still have an incentive to play σ, because if it turns out that $B(\theta, \sigma) = a_{f_3}$ then (from Proposition 1) $B(\theta, \sigma''') \neq a_{f_3}$ and consequently $\overline{U}(\theta, \sigma''') > \max\{U(\sigma', a_{f_1}), U(\sigma'', a_{f_2})\}$. In essence, if the execution of a dominated strategy can provide information about the follower preferences that will become critical in subsequent rounds of the game, one pruning heuristic might be to not prune such dominated strategy.

4.1.1 The Pruning Algorithm

When an MCTS trial starts (at the root node), the leader does not know how the follower is going to respond to *any* of its mixed strategies $\sigma \in \Sigma$, for it does not know the type $\theta \in \Theta$ of the follower that it is playing with. That is, the leader knows nothing about the sets Σ_{a_f} and anti-sets $\overline{\Sigma}_{a_f}$; $a_f \in A_f$. As the game enters subsequent rounds though, the leader collects the information about the follower responses to the leader strategies, assembles this information to infer more about Σ_{a_f} and $\overline{\Sigma}_{a_f}$; $a_f \in A_f$ and then prunes the provably dominated leader strategies that do not provide critical information to be used in later rounds of the game.

The pruning algorithm runs orthogonally to MCTS and can be applied to any MCTS node whose parent has already been serviced by the pruning algorithm. Consider one such MCTS node corresponding to a situation where the rounds $1, 2, ..., k-1$ of the game have already been played and let $\Sigma^{(k-1)} \subset \Sigma$ denote the set of leader strategies that have not yet been pruned (not to be confused with the set $E^{(k-1)}$ of leader strategies employed in rounds $1, 2, ..., k-1$ of the game). We have $\Sigma^{(0)} = \Sigma$ at the MCTS root node. Also, let $\Sigma_{a_f}^{(k-1)} \subset \Sigma_{a_f}$ and $\overline{\Sigma}_{a_f}^{(k-1)} \subset \overline{\Sigma}_{a_f}$ be the partially uncovered follower best response sets and anti-sets, inferred by the leader from its observations of the follower responses in rounds $1, 2, ..., k-1$ of the game. (Unless $|A_f| = 1$, we have $\Sigma_{a_f}^{(0)} = \emptyset$, $\overline{\Sigma}_{a_f}^{(0)} = \emptyset$; $a_f \in A_f$ at the MCTS root node.) When the leader then plays $\sigma \in \Sigma^{(k-1)}$ in the k-*th* round of the game and observes the follower best response $b \in A_f$, it constructs the sets $\Sigma^{(k)}$, $\Sigma_{a_f}^{(k)}$, $\overline{\Sigma}_{a_f}^{(k)}$; $a_f \in A_f$ as described in Algorithm 1.

Algorithm 1 starts by cloning the non-pruned action set (line 1) and best response sets (lines 2 and 3). Then, in line 4, $\Sigma_b^{(k)}$ becomes the minimal convex hull that encompasses itself and the leader strategy σ (computed e.g. using a linear program). At this point (lines 5 and 6), the algorithm constructs the best response anti-sets, for each $b' \in A_f$. In particular: $\sigma' \notin \Sigma_{b'}^{(k)}$ is added to the anti-set $\overline{\Sigma}_{b'}^{(k)}$ if there exists a vector (σ', σ'') where $\sigma'' \in \Sigma_{b'}^{(k)}$ that intersects some

Algorithm 1

Input: σ, b, $\Sigma^{(k-1)}$, $\Sigma_{a_f}^{(k-1)}$; $a_f \in A_f$

Output: $\Sigma^{(k)}$, $\Sigma_{a_f}^{(k)}$, $a_f \in A_f$

```
 1: $\Sigma^{(k)} \leftarrow \Sigma^{(k-1)}$
 2: for all $b' \in A_f$ do
 3:     $\Sigma_{b'}^{(k)} \leftarrow \Sigma_{b'}^{(k-1)}$
 4: $\Sigma_b^{(k)} \leftarrow$ ConvexHull$(\Sigma_b^{(k)}, \sigma)$
 5: for all $b' \in A_f$ do
 6:     $\overline{\Sigma}_{b'}^{(k)} \leftarrow \{\sigma' \in \Sigma \setminus \Sigma_{b'}^{(k)}$ s.t. $(\lambda\sigma' + (1-\lambda)\sigma'') \in \Sigma_{a_f}^{(k)}$
        for some $\lambda > 0$; $\sigma'' \in \Sigma_{b'}^{(k)}$ and $a_f \in A_f$; $a_f \neq b'\}$
 7: $\sigma^* \leftarrow \arg\max[\sigma' \in \Sigma_b^{(k)}]\ \{U(\sigma', b)\}$
 8: $\Sigma^{(k)} \leftarrow \Sigma^{(k)} \setminus (\Sigma_b^{(k)} \setminus \{\sigma^*\})$
 9: for all $\sigma \in \Sigma^{(k)} \setminus \cup_{a_f \in A_f} \Sigma_{a_f}^{(k)}$ and all $b \in A_f$ do
10:     if $\sigma \in \cap_{a_f \in A_f \setminus \{b\}} \overline{\Sigma}_{f_a}^{(k)}$ then
11:         goto 4
```

set $\Sigma_{a_f}^{(k)}$; $a_f \neq b$ (else, $\Sigma_{b'}^{(k)} \cup \{\sigma'\}$ would not be convex, thus violating Proposition 1). Next (lines 7 and 8), the algorithm prunes from $\Sigma^{(k)}$ all the strategies that are strictly dominated by σ^*, for which the leader already knowns the best response $b \in A_f$ of the follower. (Notice that no further information about the follower preferences can be gained by pruning these actions.) Finally, the algorithm loops (line 9) over all the non-pruned leader strategies σ for which the best response of the follower is still unknown; In particular (line 10) if $b \in A_f$ is the only remaining plausible follower response to σ, it automatically becomes the best follower response to σ and the algorithm goes back to line 4 where it considers the response b to the leader strategy σ as if it was actually observed. The pruning algorithm terminates its servicing of an MCTS node once no further actions can be pruned from $\Sigma^{(k)}$. One can then identify the leader strategies to be pruned from Equations (4, 5, 6).

5. EXPERIMENTS

5.1 Basic Checks

We first performed a series of experiments aimed at checking the validity of MCTS generated policy. One of these experiments (refer to Figure 5) is in the airport security domain of Figure 1. We set the discretization interval of leader mixed strategies to 0.25, resulting in a total of 5 leader actions. We considered a 4 stage game and ran 1,000,000 MCTS trials assuming (a) a follower whose payoffs are sampled from a uniform prior distribution and (b) a follower whose payoffs are sampled from some distribution that reflects conflicting preferences of the leader and follower agents. As can be seen in Figure 5, the resulting MCTS policies appear to conform to our intuitions. In particular, in the uniform prior distribution case, notice how the leader chooses to play [PT1=50%,PT2=50%] in the third round of the game, to effectively learn (and later take advantage of) the follower best response to [PT1=50%,PT2=50%], having learned in the earlier rounds of the game the follower responses to the leader pure strategies. Also, in the zero-sum prior distribution case, notice that when the follower is identified to prefer to AT1, the leader never chooses to play a pure strategy PT1 as this would result in Terminal 2 being left unguarded.

An illustration of typical rate-of-convergence behavior in

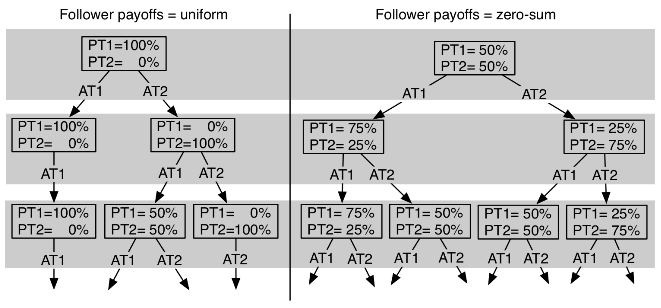


Figure 5: Policies generated by MCTS

our experiments is plotted in Figure 6. These experiments used followers with uniform random utility functions over 4 pure strategies, horizon $H = 6$, and leader discretization $\epsilon = 0.25$. The number of leader pure strategies is varied over $\{2, 3, 4\}$. We generally find clear convergence to the optimal policy and value estimate, at least in all of our small-scale studies. Note that the convergence time may not have a simple dependence on number of leader strategies, as it may also depend on specific details of the leader's utility function.

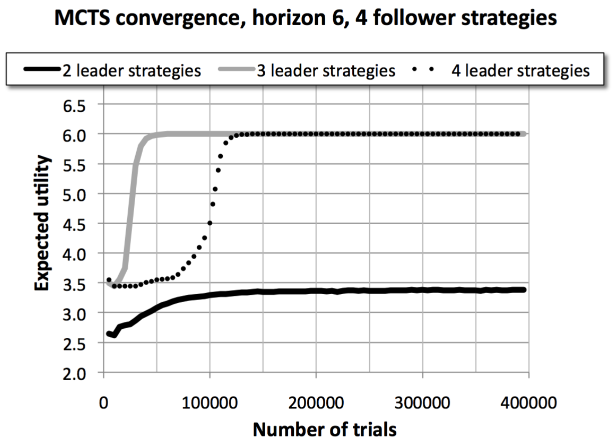


Figure 6: Value of best top-level action node vs. number of trials, illustrating typical MCTS convergence behavior.

5.2 Scaling studies without pruning

We present two scaling studies that examine how far we can push a "vanilla" version of MCTS with no built-in domain knowledge, and no capability of inference across nodes, so that the only way to estimate the value of a tree node is by explicit sampling. The first study focuses on scaling of convergence time with horizon H. We would expect MCTS convergence time to scale exponentially with H, possibly with a large base if the branching factor is large. However, certain aspects of the Bayesian Stackelberg domain could result in relatively mild scaling. First, it seems plausible that the optimal policy would consist of "exploratory" actions for the first few rounds, in order to determine the follower preferences, followed by pure exploitation actions that maximize immediate payoff in all remaining rounds. This means that it could be relatively easy for MCTS to find the best action at deep levels of the tree, as it would only need to find the action with best immediate payoff. This is easy to determine in our domain, since the payoffs are deterministic given the leader and follower actions, and the follower type is likely to be uniquely determined deep in the tree.

Results of scaling of mean convergence time with H, from $H = 2$ to $H = 16$, are shown in Table 1. We define "convergence time" as the number of trials needed to reach 99% of asymptotic optimal value. We use uniform random follower utility functions, and fix the following experiment parameters: leader strategies = 3, follower strategies = 4, and discretization = 0.25.

Note that there are 15 possible leader mixed-strategy combinations, so the branching factor per round is nominally 60, including the possible follower responses. However, we can see much more favorable scaling in Table 1 than would be implied by the nominal branching factor.

Horizon	Trials to Converge	CPU (sec)
2	15,300	13.40
4	26,100	23.53
6	42,200	38.48
8	77,500	72.60
10	132,000	127.92
12	221,000	227.79
14	340,000	515.22
16	512,000	1124.75

Table 1: Number of MCTS trials to converge and CPU runtime as a function of horizon H.

Next, we examine scaling of convergence time in a fixed experiment configuration (uniform random followers, 3 leader and 4 follower strategies, and $H = 6$) where we vary the discretization interval ϵ. Here we expect that "vanilla" MCTS could perform quite poorly, as the branching factor increases exponentially in $1/\epsilon$, and MCTS has to explicitly sample each option to acquire any information regarding its value. Results shown in Table 2 indicate that the convergence time in fact blows up rapidly once ϵ becomes small, and we were unable to obtain convergence within one million trials for $\epsilon = 1/16$. The observed poor scaling with discretization provides ample motivation for using our pruning algorithm of Section 4, whose results are reported below.

Discretization	Trials to Converge	CPU (sec)
0.5	21,100	18.22
0.25	42,200	38.48
0.125	177,000	193.56
0.0625	—	—

Table 2: Mean convergence time (in thousands of trials and CPU runtime) as a function of discretization ϵ. The 0.0625 run failed to converge within one million trials.

5.3 Results using leader strategy pruning

In our final set of experiments we switched on the pruning of the dominated leader strategies to see how it improves the performance of our algorithm. We chose the number of leader and follower pure strategies to be $M = 2$ and $N = 2$ respectively and fixed the number of game rounds to $H = 10$. Across the three experiments in Figure 7 we progressively increased the discretization accuracy of the leader mixed strategies, by *decreasing* the length of the discretization interval from 25% to 12% and finally to 6%. For each setting we then ran a sufficient number of MCTS trials to obtain convergence, and plotted on the x-axis the current trial number and on the y-axis the expected utility of the currently best leader strategy.

All three experiments in Figure 7 show that using our pruning technique results in a clear reduction in the number of trials needed to obtain convergence. As can be seen in the first graph ($\epsilon = 1/4$), with pruning switched on, the algorithm converged to within 99% of its asymptotically optimal value in less than $100,000$ trials as compared to more than $200,000$ trials needed to accomplish the same task with pruning switched off. The impact of pruning is even more pronounced in the second graph ($\epsilon = 1/8$): Here, with pruning switched on, the algorithm managed to converge to the asymptotically optimal solution after approximately $670,000$ trials. In contrast, convergence with pruning switched off required more than 1.2 million trials, i.e., pruning reduced the required number of trials by over $500,000$. Finally, in the third graph in Figure 7, the increased accuracy of the discretization of the leader mixed strategies ($\epsilon = 1/16$) resulted in a game tree with 2^{50} nodes. The algorithm with pruning converged in $\sim$ 4.2 million trials, representing a savings of 1M trials compared to the 5.2 million needed for convergence without pruning.

Of course, the number of trials needed for convergence is not the only relevant performance metric in these experiments, since the pruning technique entails greater computational overhead per trial. If the CPU cost of the pruning calculations were to exceed the savings in number of required trials, the pruning technique would not yield a net win in terms of wall-clock run time. Fortunately, our implementation of pruning limits the CPU cost of pruning to at most 50% of total simulation CPU time: we run pruning and MCTS search on two independent threads, with the pruning thread taking up to 50% of the cycles of a single CPU, and the search thread taking the remaining cycles. To obtain a fair comparison, the corresponding experiments without pruning also utilized two threads, each performing independent MCTS trials. As seen below in Table 3, the savings in number of trials more than compensates for the greater CPU overhead of the pruning technique. We observe a progressive increase in wall-clock time savings via pruning, from 25 seconds (157-132) at $\epsilon = 1/4$, to 94 seconds (637-540) at $\epsilon = 1/8$, to 277 seconds (2903-2626) at $\epsilon = 1/16$.

Discretization	Pruning Time	No-Pruning Time
0.25	132	157
0.125	540	637
0.0625	2626	2903

Table 3: Mean wall-clock time to converge, in seconds, as a function of discretization ϵ, in pruning vs. no-pruning experiments of Figure 7.

6. CONCLUSIONS

We presented the first-ever study of repeated Bayesian Stackelberg games where the objective is to maximize the leader's cumulative expected payoff over the rounds of the game. The optimal policies in such games must make intel-

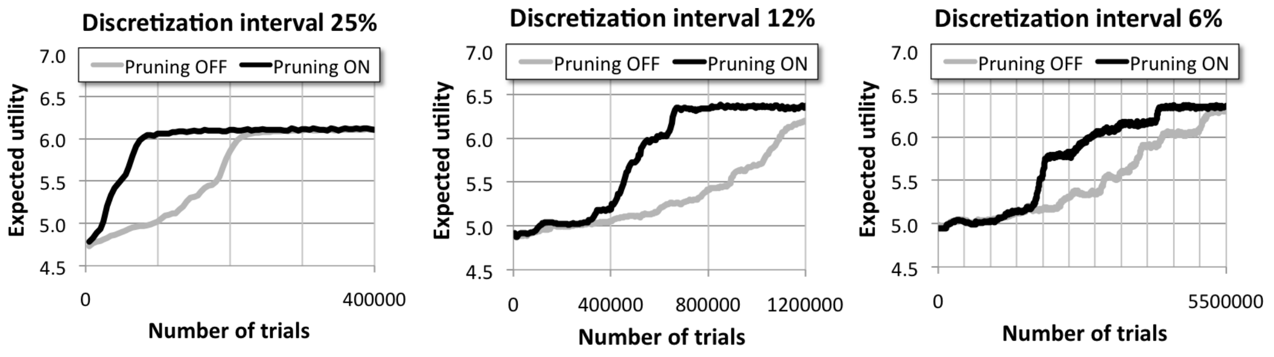


Figure 7: Impact of pruning of dominated leader strategies on the efficiency of MCTS

ligent tradeoffs between actions that reveal information regarding the unknown follower preferences, and actions that aim for high immediate payoff. We proposed an innovative approach to solve for such optimal policies, based on Monte Carlo Tree Search combined with an original method for pruning dominated leader strategies.

Our results show ample promise that the approach will be able to tackle numerous problems in real-world security domains. The points of particular promise that we see are: (1) MCTS trials based on sampling from the follower distribution appears to offer a highly efficient means of approximating follower posteriors over all possible history paths, insofar as they contribute to the optimal policy. (2) MCTS scaling is very manageable with respect to the number of rounds to be played. (3) Our pruning method addresses a major limitation of simple MCTS which scales poorly in the case of fine-grained discretization of leader mixed strategies.

Our findings draw support from the recent work by Silver and Veness [17] which obtained excellent scaling results in applying MCTS to solving large scale POMDPs. This suggests to us that the advantages of MCTS that we found in Bayesian Stackelberg Games may extend to more general classes of imperfect information games.

One of our future research studies will focus on generalizing the model to repeated Bayesian Stackelberg games where the follower also behaves strategically. In this more general case there might be scenarios where MCTS convergence may be much more difficult than what we found in the non-strategic case. This has been observed in recent work by Ramanujan et al. [15] which identified *soft traps.* We believe that such traps do not occur in our current formulation as the leader always gains information about the follower preferences. However, how to avoid such traps when facing a strategic follower player is a worthy topic of investigation.

7. ACKNOWLEDGMENTS

This work was supported in part under the DARPA GALE project, contract No. HR0011-08-C-0110.

8. REFERENCES

[1] T. Alpcan and T. Basar. A game theoretic approach to decision and analysis in network intrusion detection. In *42nd IEEE Conf. on Decision and Control*, 2003.

[2] P. Auer, N. Cesa-Binachi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.

[3] H. Finnsson and Y. Björnsson. Simulation-based approach to general game playing. In *Proc. of AAAI-2008*, pages 259–264, 2008.

[4] D. Fudenberg and J. Tirole. *Game Theory.* MIT Press, 1991.

[5] S. Gelly et al. Modification of UCT with patterns in Monte-Carlo Go. Technical Report 6062, INRIA, 2006.

[6] M. Jain et al. Security games with arbitrary schedules. In *AAAI*, 2010.

[7] C. Kiekintveld, J. Marecki, and M. Tambe. Approximation methods for infinite bayesian stackelberg games: Modeling distributional payoff uncertainty. In *AAMAS*, 2011.

[8] L. Kocsis and C. Szepesvari. Bandit based Monte-Carlo planning. In *15th European Conf. on Machine Learning*, pages 282–293, 2006.

[9] J. Letchford, V. Conitzer, and K. Munagala. Learning and approximating the optimal strategy to commit to. In *Symp. on the Algorithmic Decision Theory*, 2009.

[10] K. Nguyen and T. Basar. Security games with incomplete information. In *IEEE Intl. Conf. on Communications*, 2009.

[11] P. Paruchuri et al. An efficient heuristic approach for security against multiple adversaries. In *AAMAS* , 2007.

[12] P. Paruchuri et al. Playing games with security: An efficient exact algorithm for Bayesian Stackelberg games. In *AAMAS*, 2008.

[13] J. Pita et al. Deployed ARMOR protection: The application of a game-theoretic model for security at the the LAX airport. In *AAMAS Industry Track*, 2008.

[14] J. Pita et al. Effective solutions for real-world stackelberg games: When agents must deal with human uncertainties. In *AAMAS*, 2009.

[15] R. Ramanujan, A. sabharwal, and B. Selman. Understanding sampling style adversarial search methods. In *Proceedings of UAI-2010*, 2010.

[16] R. Segal. On the scalability of parallel UCT. In H. van den Herik et al., editors, *Computers and Games*, volume 6515 of *LNCS*, pages 36–47. Springer Berlin / Heidelberg, 2011.

[17] D. Silver and J. Veness. Monte-Carlo Planning in Large POMDPs. In *Advances in Neural Information Processing Systems 22*, 2010.

Repeated zero-sum games with budget

Troels Bjerre Sørensen
University of Warwick
CV4 7AL, United Kingdom
trold@dcs.warwick.ac.uk

ABSTRACT

When a zero-sum game is played once, a risk-neutral player will want to maximize his expected outcome in that single play. However, if that single play instead only determines how much one player must pay to the other, and the same game must be played again, until either player runs out of money, optimal play may differ. Optimal play may require using different strategies depending on how much money has been won or lost. Computing these strategies is rarely feasible, as the state space is often large. This can be addressed by playing the same strategy in all situations, though this will in general sacrifice optimality. Purely maximizing expectation for each round in this way can be arbitrarily bad. We therefore propose a new solution concept that has guaranteed performance bounds, and we provide an efficient algorithm for computing it. The solution concept is closely related to the Aumann-Serrano index of riskiness, that is used to evaluate different gambles against each other. The primary difference is that instead of being offered fixed gambles, the game is adversarial.

Categories and Subject Descriptors

I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems

General Terms

Algorithms, Economics, Theory

Keywords

Game playing, Game theory

1. INTRODUCTION

Game theory has often been used to prescribe good behavior in strategic interactions, and to make predictions on how participants will behave in interaction with one another. This is traditionally done by isolating a particular interaction of interest, and then modelling the interaction mathematically. The constructed model is then analyzed separately from the rest of the system, and the resulting analysis is then translated back into the system where the interaction fits in. For this approach to be successful, the right objectives must be derived from the system surrounding the interaction; otherwise the analysis will likely draw incorrect conclusions. In this paper, we examine such a situation, where a fixed finite zero-sum game is played repeatedly under a budget. The overall goal for each player is to win all the money the opponent has, and not run out of money in the process. A simple approach to analyzing this repeated game would be to analyze the zero-sum game as if it was only played once, with the objective of winning the most money in that single play. This, however, can lead to disastrous results, as we shall see in Section 4. In this paper, we show how one can analyze the underlying zero-sum game with respect to a more suitable objective.

As a motivating example, let us look at what happens when we offer a player the chance to double the payoffs of a zero-sum game. Given any zero-sum game G with payoffs in $\{-1, 1\}$ and value $v \neq 0$. Amend G by giving Player 1 the option of doubling the outcome of the game, before the game is played. If Player 1 does so, the outcomes will be in $\{-2, 2\}$ instead, but the rest of the game is unchanged. Any risk neutral Player 1 would clearly double the game, if and only if the value of the game is positive; the doubling does not change optimal strategies, but it doubles the expected value. What is perhaps more surprising is that the situation is reversed if the game has to be repeated until one player is broke; i.e., Player 1 would only double if the value was negative. To see this, first observe that two players playing the undoubled game optimally for some total amount of money, C, is simply a random walk on a line of length C. The walk moves one step to the right with probability $p = v/2 + 1/2$, and one step to the left with probability $(1 - p)$. Starting from point c_1 (being the amount of money Player 1 has), the probability of reaching the right-most point (where Player 1 has won all the money) before the left-most point is exactly

$$Pr[\text{Player 1 wins undoubled}] = \frac{\alpha^{c_1} - 1}{\alpha^{C} - 1} \tag{1}$$

where $\alpha = \frac{1-p}{p}$. Playing the doubled game is essentially the same as playing the undoubled game for half as much money. Assume for simplicity that C and c_1 are both even. This means that the probability of Player 1 winning by doubling every game is

$$Pr[\text{Player 1 wins doubled}] = \frac{\alpha^{c_1/2} - 1}{\alpha^{C/2} - 1} \tag{2}$$

This probability is greater than that (1) if and only if $\alpha > 1$, which happens exactly when $v < 0$. Thus, Player 1 should

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

double the game, if and only if he has *negative* expectation in the individual rounds of the game.

This example serves to show that if we attempt to derive good strategies for the repeated game by trying to maximize the expected outcome of the individual rounds, we will get suboptimal results. In Section 4, we will show that this suboptimality can be arbitrarily large.

1.1 Related research

In a recent paper, Miltersen and Sørensen [13] computed near optimal strategies for a full scale two-player poker tournament. The tournament format fits the description of a game being played repeatedly for a budget, but their game was different for each round; the variant of poker allowed for an all-in, which depends on how much money each player has. They concluded that simply maximizing the amount of chips won in each round was slightly worse than maximizing the probability of winning the tournament. In contrast, the present paper shows that it can be much worse to maximize the expected gain in each round. Furthermore, our results are about general zero-sum games, and not poker specific.

The conceptual ancestor of the contribution of this paper is the Aumann-Serrano index of riskiness [1], which is used to compare different gambles against each other. The main difference is that the Aumann-Serrano index of riskiness is not in an adversarial setting. The Aumann-Serrano index of riskiness of a stochastic variable X is defined as the unique γ such that $E[exp(-X/\gamma)] = 1$. Expressed in these terms, our contribution is to compute the strategy that has the most favorable Aumann-Serrano index of riskiness on the outcome of the game.

1.2 Structure of the paper

The rest of the paper is structured as follows. In Section 2 we introduce the formal model of the games we are discussing in this paper. In Section 3, we review existing theory that provide exact optimal strategies for the games, and discuss why this is not a feasible approach. In Section 4, we show why maximizing expectation in each round can be arbitrarily far from optimal. In Section 5, we describe the main contribution of the paper in the form of a new solution concept and an algorithm for computing it. In Section 6, we derive a bound on the performance of the introduced solution concept. In Section 7, we apply the theory to the game Kuhn poker with a budget. In Section 8, we provide a way to estimate the parameter of the introduced solution concept for games that are too large to repeatedly solve. In Section 9, we discuss two natural extensions of the theory. In Section 10 we compare the introduced solution concept to existing concepts, and discuss future research.

2. MODEL

In this section, we will formalize the model we are using in this paper. First we need some notation and terminology from classic game theory. Details can be found in any introductory textbook on game theory. The underlying game to be played is given as a finite zero-sum game:

DEFINITION 1 (FINITE ZERO-SUM GAME). *A finite zero-sum game is given by $m \times n$ matrix A with integer entries. It is played by Player 1 and Player 2 simultaneously choosing a row i and a column j respectively, after which Player 2 pays A_{ij} to Player 1.*

The definition above has a non-standard assumption that the outcome of the finite zero-sum games are integer. This is only to make analysis easier, and everything in this paper can be done with rational valued outcomes as well, as is discussed in section 9.

The players can use *mixed strategies*, that are probability distributions over rows and columns respectively. For zero-sum games, there is a well defined value that each player can guarantee himself, and the associated strategies that provide this guarantee:

DEFINITION 2 (MINIMAX VALUE AND STRATEGIES). *The minimax value of a finite zero-sum game given by the matrix $A \in \mathbb{Z}^{m \times n}$ is:*

$$val(A) = \max_{x \in \Delta^m} \min_{y \in \Delta^n} x^\top Ay = \min_{y \in \Delta^n} \max_{x \in \Delta^m} x^\top Ay$$

The minimax strategies for Player 1 are the maximizing x's in the expression above:

$$\operatorname*{argmax}_{x \in \Delta^m} \min_{y \in \Delta^n} x^\top Ay$$

Likewise, the minimax strategies for Player 2 are

$$\operatorname*{argmin}_{y \in \Delta^n} \max_{x \in \Delta^m} x^\top Ay$$

In the setting we are examining in this paper, the game is played repeatedly between two players, each starting with some amount of money, c_1 and c_2. The game progresses over a number of rounds, each of which is a play of a finite zero-sum game. After a round is played, money changes hands according to the strategies (i, j) chosen by the two players, and the game continues with $c_1 = c_1 + A_{ij}$ and $c_2 = c_2 - A_{ij}$, unless either player has run out of money. If that happens the game ends, and the player who is out of money has lost the game, and his opponent has won. Notice that the total amount of money stays constant throughout the repeated game. Denote this constant by $C = c_1 + c_2$. It is of course not possible to win more money from the other player than he has, so the last round might not have full payment off all of A_{ij}. Each player naturally wants to maximize the probability of ending up with all the money, thereby winning the whole game. As in [13], this is not necessarily the same as maximizing the amount of money in each round. In the next section, we will quantify exactly how bad this can be.

Before we can continue, we need to handle certain special cases of games.

DEFINITION 3 (DEGENERATE GAME). *We call a game degenerate, if either player has a strategy that never loses any money against any strategy of the opponent. We also call a game degenerate, if it has equilibria with deterministic outcome 0.*

If the first is the case, then one of the players doesn't have any chance of winning the game, if his opponent tries to prevent it. This is not the same as the opponent always being able to win, and as such, the objective of the game is not necessarily clear; it depends on whether infinite play is truly an acceptable outcome. We have chosen to sidestep this complication, as it is caused by a degenerate input game. For the same reason, we will assume that the game does not have equilibria with deterministic outcome 0, as this also opens the possibility of infinite play.

3. EVERETT'S RECURSIVE GAMES

In this section we will describe how to play the repeated game optimally, and discuss why this is often infeasible. If the total amount of money is known beforehand, the game can be modelled and solved as a *recursive game*, introduced by Everett [7]. These recursive games should not be confused with the largely unrelated recursive games by Etessami and Yannakakis [6]. Everett's recursive games is a generalization of *concurrent reachability games* [5] and of *simple stochastic games* [3, 4]. A recursive game consists of a set of *game elements*, each of which are finite zero-sum games, with the added possibility of a outcome being a reference to another game element. If a normal outcome is reached, the game ends with the associated value as the zero-sum outcome. If one of the special outcomes is reached, the play must continue at the referenced game element. This opens up the possibility of the game never ending, which we assign the value 0.

This model fits the repeated game setting in the following way. There will be a game element for every possible division of money between the two players, and each game element will be indexed by how much money Player 1 has. The outcomes of each game element will be references to the neighboring game elements, such that outcome A_{ij} from game element indexed c_1 will be a reference to game element indexed $c_1 + A_{ij}$.

Everett proved that these games can be played ϵ-optimally using *stationary strategies.* A stationary strategy consists of one strategy per game element, and it is played by always using the strategy associated with the current game element. This means that there is nothing to be gained for a player by remembering what game elements have been visited prior to playing a particular game element.

Everett showed that a *critical value* can be assigned to each game element, similar to the minimax value of a finite zero-sum game, such that each player can guarantee an expected outcome arbitrarily close to the assigned value, if the play was started at that game element. The vector of the critical values for all game elements is called the *critical vector.*

If we assign value 0 to game element 0, and value 1 to game element C, the critical value of a game element will be exactly the probability that Player 1 can guarantee himself of winning the repeated game.

In general, there is no easy way to check whether a given vector is an upper bound to the critical vector of a given recursive game. However, Everett gave a property that can be checked in polynomial time that would hold for a subset of the upper bounds and another property that would hold for a subset of the lower bounds. Before we can formally state the property, we need to define the value mapping:

Definition 4 (Value vector and value mapping). *Let G be a recursive game with n game elements. A value vector $\vec{v} \in \mathbb{R}^n$ for G is a vector with one value for each game element of G. The value mapping $\mathbb{M} : \mathbb{R}^n \to \mathbb{R}^n$ of G, mapping value vectors to value vectors, is the minimax evaluation of each game element, where the non-terminal outcomes have been replaced with the values given by the input vector.*

The properties rely on the following relations among value vectors:

$$\vec{u} \succeq \vec{v} \Leftrightarrow \left\{ \begin{array}{ll} \vec{u}_i > \vec{v}_i & \text{if } \vec{v}_i > 0 \\ \vec{u}_i \geq \vec{v}_i & \text{if } \vec{v}_i \leq 0 \end{array} \right\} \forall i$$

$$\vec{u} \preceq \vec{v} \Leftrightarrow \left\{ \begin{array}{ll} \vec{u}_i < \vec{v}_i & \text{if } \vec{v}_i < 0 \\ \vec{u}_i \leq \vec{v}_i & \text{if } \vec{v}_i \geq 0 \end{array} \right\} \forall i$$

Everett proved the following Theorem:

Theorem 5 (Everett, 1957).
If $\mathbb{M}(\vec{v}) \succeq \vec{v}$, then v is a lower bound on the critical vector. Furthermore, the stationary strategy for Player 1 obtained by finding the optimal strategy in each game element, with arcs to other game elements replaced by the corresponding values in $\vec{v}$, has guaranteed expected payoff at least *$\vec{v}_g$ for play starting in g. If $\mathbb{M}(\vec{v}) \preceq \vec{v}$, then $\vec{v}$ is an upper bound on the critical vector.*

In short, if the value mapping increases the value of each entry of a value vector, then the new values is a lower bound on the critical vector. It is this property we will use later in the paper to give performance guarantees on the introduced solution concept.

For general recursive games, the only known algorithm for computing the exact critical vector is that of Hansen et.al. [9]. This algorithm runs in time doubly exponential in the size of the game, and outputs the values as algebraic numbers in isolating interval representation. However, we can approximate the critical vector efficiently using value iteration. This approach does not work for general recursive games, as shown by Everett, but as discussed below, it works for our special case.

In this context, value iteration means repeatedly applying the value mapping to a value vector, until it converges. An easy way to detect convergence is to run two value iterations, one starting from a trivial upper bound and the other starting from a trivial lower bound. In our case, as the values are probabilities, a vector of 0s would serve as a lower bound, while a vector of 1s would work as an upper bound. Notice that $\mathbb{M}(\vec{v})$ is monotone in $\vec{v}$, so if $\vec{v}$ is an upper (resp. lower) bound to the critical vector, then so is $\mathbb{M}(\vec{v})$. Thus, if the two vectors are close after a number of iterations, then we have a good approximation to the critical vector. Now we only need to show that the two vectors will in fact get close. Notice that the value iteration on the lower bound after T steps corresponds exactly to the time limited game, where Player 2 is declared the winner if the game doesn't end naturally before T steps. Similarly with Player 1 for the upper bound. Since the game is non-degenerate, both Players have positive probability of winning money from any given game element. Thus, the probability of the game not having ended after T steps goes to 0 as T goes to infinity. For a more thorough analysis of this type of algorithms, see [8].

While this approach gives a provably optimal strategy for the repeated game, it might not be a feasible approach for several reasons. First of all, the explicit modelling requires one game element for every non-degenerate division of money between the two players, which is $C - 1$ game elements. This number might be prohibitively large from a computational point of view, as more game elements requires more work. It also has the problem that each player must remember one strategy per game element, as the strategies in general will be different. Finally, it might be that a player does not know how much money his opponent has, in which case the explicit modelling given above is not possible.

Definition 6 (Oblivious strategy).
A positional strategy is oblivious *if it associates the same strategy to all game elements.*

Using an oblivious strategies, the player needs only remember a single strategy, but it will in general not be optimal in the recursive game. To the best of our knowledge, there is no algorithm for computing the best oblivious strategy for this setting.

4. FAILURE OF MINIMAX

In this section, we will prove why simply maximizing immediate outcome of each round can be arbitrarily far from optimal. We will do this by explicit construction of a finite zero-sum game with a unique minimax strategy that wins with probability zero, where the optimal strategy would win with probability arbitrarily close to 1. Let $n \geq 4$ be an even number. Now construct $A \in \mathbb{Z}^{2n \times n}$ in the following way:

$$A_{ij} = \begin{cases} -1 & \text{if } i = j \\ 0 & \text{if } i \neq j \wedge i \leq n \\ 1 & \text{if } i > n \wedge [(i+j) \bmod n \leq n/2 - 1] \\ -1 & \text{otherwise} \end{cases}$$

Both players have unique minimax strategies; Player 2 uniformly mixes over all n columns, while Player 1 uniformly mixes over the n first rows. The one-round game thus has value $-1/n$. However, the row player would never win the repeated game using this strategy, as the strategy never wins any money. If the row player instead uniformly mixed over the n last rows, he would win a coin with probability $1/2 - 1/n$, and lose a coin with probability $1/2 + 1/n$. Player 2's minimax strategies are still a best response, even in the repeated setting. The expected gain in each round is lowered to $-2/n$, but the probability of winning something is now just below 1/2 instead of 0. If the repeated game is started from $(c_1 = k-1, c_2 = 1)$, the probability of Player 1 winning with the second strategy will be almost $1 - 1/k$, while the first strategy will win with probability 0. Pick n and k large enough, and the minimax strategies turn an almost sure win into a certain loss.

5. RISKINESS OF A GAME

In this section we will describe the main contribution of the paper. In short, we show how to find the right exponential utility function for a given game, such that minimax strategies with respect to that utility function will have good guaranteed bounds on the performance in the repeated game. Utility functions are functions from outcomes to real values, describing a player's satisfaction with a particular outcome. They are commonly used to explain how both the seller and the buyer of insurance policies can be satisfied by the transaction, even though the underlying transaction is zero-sum. A risk-averse (concave utility function) insuree is happy to pay an insurance premium that is more than the expected loss, in exchange for less variance of the outcome. Similarly, a risk-seeking gambler (convex utility function) will buy lottery tickets, even though the expected outcome is lower than the price of the ticket. For our purpose, the players want to maximize the probability of winning the tournament, which in general is not linear in the money won in each round. We therefore want to find the right function to serve as a proxy for the probability of winning the tournament. Let us first define the exponentiation of a game:

Definition 7 (Exponentiation of a game).
Given a zero-sum game $A \in \mathbb{Z}^{m \times n}$ and a positive constant α, define A^α to be

$$A^\alpha_{ij} = \begin{cases} \frac{\alpha^{A_{ij}} - 1}{\ln \alpha} & , \textit{if } \alpha \neq 1 \\ A_{ij} & , \textit{if } \alpha = 1 \end{cases}$$

Notice that the entries of A^α are continuous in α . If $\alpha = 1$, then $A^\alpha = A$, corresponding to the players being completely risk neutral. If $\alpha > 1$, the utility function is convex for Player 1 and concave for Player 2, making Player 1 risk seeking, while Player 2 will be risk averse. The situation is the opposite for $\alpha < 1$. We are now looking for a suitable α for the game at hand.

Proposition 8.
Given a non-degenerate zero-sum game $A \in \mathbb{Z}^{m \times n}$, there exists an α^* such that $val(A^{\alpha^*}) = 0$.

Proof sketch. Since all entries of A^α are continuous functions of α, we know that $val(A^\alpha)$ is also continuous in α. Since A is non-degenerate, Player 1 has a strategy that guarantees at least some fixed strictly positive probability of a positive outcome. As all positive entries of A^α approach ∞ as $\alpha \to \infty$, and all negative entries approach 0, we have the $val(A^{\alpha_{hi}}) > 0$ for some sufficiently large α_{hi}. Likewise, Player 2 has a strategy that guarantees at least some fixed strictly positive probability of a negative outcome. As all positive entries of A^α approach 0 as $\alpha \to 0$, and all negative entries approach $-\infty$, we have the $val(A^{\alpha_{lo}}) < 0$ for some sufficiently small α_{lo}. Combined with continuity of $val(A^\alpha)$ in α, the intermediate value theorem gives us that there exists some α^* such that $val(A^{\alpha^*}) = 0$. $\square$

Proposition 9.
Given a non-degenerate zero-sum game $A \in \mathbb{Z}^{m \times n}$, there is only one α^* such that $val(A^{\alpha^*}) = 0$.

Proof sketch. We need to prove that $val(A^{\alpha^*})$ is strictly monotone in α, from which the proposition follows. Given $\alpha_1 < \alpha_2$, we must prove that $val(A^{\alpha_1}) < val(A^{\alpha_2})$. Notice first that the payoff of all strategy combinations strictly increases in α, unless they result in a deterministic outcome of 0. Let x be the strategy for Player 1 that guarantees $val(A^{\alpha_1})$ in A^{α_1}. Unless Player 2 has a response that guarantees a deterministic outcome of 0, the value of all responses of Player 2 will be strictly higher in A^{α_2} than in A^{α_1}, and Player 1 can thus use x to get a higher value in A^{α_2} than in A^{α_1}, from which it follows that $val(A^{\alpha_1}) < val(A^{\alpha_2})$. If $val(A^{\alpha_1}) < 0$, Player 2 has no desire to use such a 0-strategy, but the payoff of all other strategies against x in A^{α_2} is higher than in A^{α_1}. If $val(A^{\alpha_1}) > 0$, Player 2 does not have a 0-strategy, and therefore $val(A^{\alpha_1}) < val(A^{\alpha}_2)$. The only case left is if $val(A^{\alpha_1}) = 0$. As the base game A does not have equilibria with deterministic outcome 0, the outcome of A^{α_1} cannot be deterministic outcome 0, therefore $val(A^{\alpha_2}) > val(A^{\alpha_1})$. $\square$

The assumption that A does not have equilibria with deterministic outcome 0 is crucial for the uniqueness of α^*.

Without this assumption, $val(A^\alpha)$ is only weakly monotone in α. To properly handle such degenerate games with multiple such α^*, we need the following slightly more general definition of the suitable value of α^*.

Definition 10 (Riskiness of a game). *Given a zero-sum game $A \in \mathbb{Z}^{m\times n}$, the riskiness for Player 1 in A is the largest α^* such that $val(A^{\alpha^*}) = 0$.*

If the game is non-degenerate, the is only one A^* satisfying the condition. If that is the case, we will simply call it the riskiness of the game.

Definition 11 (risk-aware strategies). *Given a non-degenerate zero-sum game $A \in \mathbb{Z}^{m\times n}$, the risk-aware strategies of A are the minimax strategies of A^{α^*}, where α^* is the riskiness for Player 1 in A.*

Even with just weak monotonicity, we can compute the riskiness for Player 1 using the algorithm given in Algorithm 1.

Algorithm 1 Computes risk-aware strategies

```
$\alpha_{hi} \leftarrow 1$
$\alpha_{lo} \leftarrow 1$
while $val(A^{\alpha_{lo}}) > 0$ do
    $\alpha_{lo} \leftarrow \alpha_{lo}/2$
end while
while $val(A^{\alpha_{hi}}) \leq 0$ do
    $\alpha_{hi} \leftarrow \alpha_{hi} * 2$
end while
while $\alpha_{hi} - \alpha_{lo} > \epsilon$ do
    $\alpha \leftarrow (\alpha_{lo} + \alpha_{hi})/2$
    if $val(A^\alpha) > 0$ then
        $\alpha_{hi} \leftarrow \alpha$
    else
        $\alpha_{lo} \leftarrow \alpha$
    end if
end while
return minimax strategies of $A^{\alpha_{hi}}$
```

6. PERFORMANCE GUARANTEE

In this section, we will show what performance guarantees we get in the repeated game, if both players have a finite budget. This is done by expressing value vectors that satisfy Everett's conditions, proving that they are true bounds on the critical vector, and that the computed strategies have the promised performance.

Theorem 12. *Given a game $A \in [-min; max]^{M\times N}$ with riskiness α, using the risk-aware strategy will guarantee Player 1 a winning probability of:*

$$\frac{\alpha^{c_1} - 1}{\alpha^{c_1+c_2+max-1} - 1}$$

when the game is started from money division (c_1, c_2).

Proof. We need to prove that the values vector described above satisfies Everett's lower-bound condition. For all *internal* game elements, with indices in $[min; C - max]$, we can use the following argument. Given a fixed game element indexed c_1, the values assigned to the game elements around it are:

$$\cdots \frac{\alpha^{i-1} - 1}{\alpha^{C+max-1} - 1}, \frac{\alpha^{i} - 1}{\alpha^{C+max-1} - 1}, \frac{\alpha^{i+1} - 1}{\alpha^{C+max-1} - 1}, \cdots$$

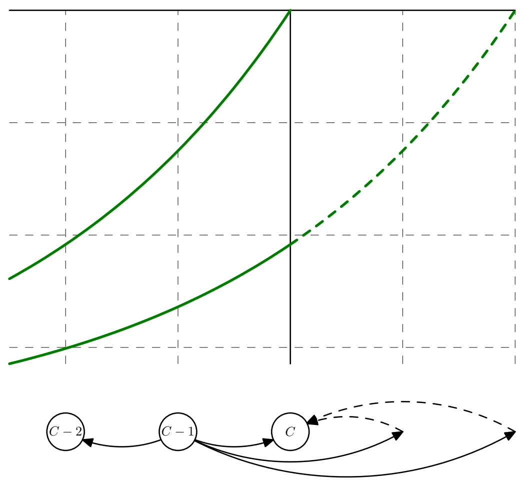


Figure 1: Shifting the value vector to satisfy Everett's condition for high indexed game element.

Optimal strategies are not changed by positive affine transformations of the utility function. Notice that if we multiply the neighborhood values by the positive constant

$$\frac{\alpha^{C+max-1} - 1}{\alpha^i \ln \alpha}$$

and add the constant

$$\frac{\alpha^{-i} - 1}{\ln \alpha}$$

the neighborhood becomes

$$\cdots \frac{\alpha^{-1} - 1}{\ln \alpha}, \frac{\alpha^{0} - 1}{\ln \alpha}, \frac{\alpha^{1} - 1}{\ln \alpha}, \cdots$$

which is exactly the exponentiated game with parameter α. This exponentiated game has positive value for all $\alpha > \alpha^*$, implying that the value mapping on the vector given in Theorem 12 increases the value of game elements with indices in $[min; C - max]$.

The outlying game elements, where either player might not have enough money to pay A_{ij}, for some (i, j), we need to argue differently. For game elements with indices in $[1; min - 1]$, some of the low values in the neighborhood is rounded up to 0, compared to what the exponentiated game looks like. But since we are increasing the value of some game elements in the neighborhood, Everett's condition will still hold, as the value mapping is monotone. The situation is different for the indices $[C - max + 1; C - 1]$, as they have neighborhoods extending beyond game element C. If we were to fit the exponentiated utility function in with $val_C = 1$, we would have to round the value down of the higher payoffs, thereby possibly violating Everett's condition. We therefore have to shift the value vector such that the non-existing game elements indexed $C + max - 1$ gets value 1, as shown in figure 1. Fitting the exponential function to the points $(0, 0)$ and $(C + max - 1, 1)$, we get exactly the expression in Theorem 12. By definition of the riskiness of Player 1, we now have that Everett's first condition is satisfied for all $\alpha > \alpha^*$ for all game elements, and we therefore have the limit as a lower bound. □

We can also use the same expression to give an upper bound on the performance, not just of these strategies, but of any strategy; even non-oblivious.

THEOREM 13. *Given a non-degenerate game with matrix $A \in [-min; max]^{m \times n}$ with riskiness α, no strategy for Player 1 can guarantee more than*

$$\frac{\alpha^{c_1+min-1}-1}{\alpha^{C+min-1}-1}$$

in the repeated game starting from division (c_1, c_2).

PROOF OF THEOREM 13. To prove this, we need a simple observation about the game from the opponents point of view, namely that the riskiness is inverted for the other player, i.e.,

$$risk(A) = risk(-A^\top)^{-1}$$

Combining this with Theorem 12 we get a strategy for Player 2 that wins with probability

$$\frac{\alpha^{-c_2}-1}{\alpha^{-C-min+1}-1}$$

when the game is started from division (c_1, c_2). Since at most one player can win, Player 1 cannot guarantee a higher probability of winning than 1 minus the guarantee Player 2 has.

$$\begin{aligned} Pr[\text{Player 1 win}] &\leq 1 - \frac{\alpha^{-c_2}-1}{\alpha^{-C-min+1}-1} \\ &= \frac{\alpha^{-C-min+1}-\alpha^{-c_2}}{\alpha^{-C-min+1}-1} \\ &= \frac{\alpha^{c_1+min-1}-1}{\alpha^{C+min-1}-1} \end{aligned}$$

□

We can use the upper bound to give a bound on how far from optimal the risk-aware strategies are. Assume for simplicity that $min = max$, i.e., that that most negative outcome is minus the most positive outcome of the game. The difference between the upper and lower bounds is greatest at game element C-1 when $\alpha > 1$, and at game element 1 when $\alpha < 1$. Assume wlog the latter is the case:

$$\text{Gap} = \frac{\alpha^{1+min-1}-1}{\alpha^{C+min-1}-1} - \frac{\alpha^{1}-1}{\alpha^{C+max-1}-1} = \frac{\alpha^{min}-\alpha}{\alpha^{C+min-1}-1}$$

Notice that the gap is 0 when $min = max = 1$. That is, the risk-aware strategies are optimal for games with outcomes in $\{-1, 0, 1\}$. The gap in the general case approached $\alpha - \alpha^{min}$ for $C \to \infty$ for $\alpha < 0$.

7. EXPERIMENTS

As an example of how well the risk-aware strategies perform, let us look at the game of Kuhn poker [12]. The game is a heavily simplified version of poker, played between two players. The rules are as follows. A deck of three cards (King, Queen and Jack) is shuffled, and the two players receive one card each. Both players put one coin in the pot as an ante. The players now use normal poker betting protocol to decide whether to bet an additional coin, i.e., Player 1 can check or bet. If he bet, Player 2 can either call or fold. If Player 1 checked, Player 2 can either check or bet. In case Player 2 bets, Player 1 has to decide whether to call or fold. If a player folds, he forfeits the hand and loses the ante he paid in the beginning of the hand. If neither player folds, the hidden cards are revealed, and the higher card wins the ante and the bets (if any).

The game has a unique equilibrium:

- Player 1 bets with King, checks with Queen, and bets (bluffs) with probability 1/3 with Jack. If Player 2 bets, Player 1 will call with probability 2/3 with Queen, and always folds with Jack.
- If Player 1 checks, Player 2 uses same strategy as Player 1 does for the first move. If Player 1 bets, Player 2 calls with King, calls with probability 1/3 with Queen, and always folds with Jack.

The game has value $-1/18$, i.e., Player 1 is expected to lose a little every round. Using the algorithm outlined earlier in the paper, we compute the riskiness of the game to be $\alpha \approx 1.062$. Solving the game exponentiated with this α, we find that the optimal strategy has changed in the following way:

- Player 1 increases the probability of bluffing with Jack to 37.6%, but slightly lowers the probability of calling with Queen to 64.6%.
- Player 2 lowers the probability of bluffing with Jack to 29.5%, but increases the probability of calling with Queens to 37.3%.

We can evaluate the two pairs of strategies against an optimal counter strategy in the following way. If we fix the strategy of one player to the strategy we want to evaluate, the other player is left with a one-player game, which we can easily solve as a Markov Decision Process; simply do value iteration with only one player. The values of these counter strategies are given in Figures 2, 3, and 4.

For larger values of C, the minimax performance gap grows to around 20%, while the risk-aware performance gap falls to less than 3%. This corresponds well with the theoretical bound on the performance gap of 5.5%, calculated using the difference between the upper and lower bound in the previous section.

8. ESTIMATING RISKINESS

Some games are so large that even solving them once requires months of computation [10] and it is therefore practically impossible to solve it repeatedly in order to do binary search and compute the riskiness of the game. It would therefore be useful to estimate the riskiness of the game beforehand, and then only compute minimax strategies of the exponentiated game once with the estimated riskiness. This can be done by observing that two fixed strategies played obliviously against each other results in a random walk on line of possible divisions of money between the players, just as in the introductory example. This process can be approximated by a Wiener process with drift, if we know just a little about the typical play in the game. The following theorem can be found in any textbook on stochastic processes:

THEOREM 14. *A Wiener process with parameter σ^2 and drift μ on a line of length C, starting at point i has probability of reaching the right endpoint before the left equal to*

$$\frac{\alpha^{i}-1}{\alpha^{C}-1}$$

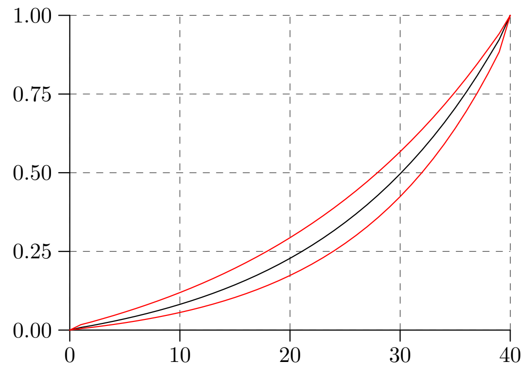


Figure 2: Upper line is the value of Player 1's best response to Player 2's minimax, lower line is value of Player 2's best response Player 1's minimax, middle line is the critical vector

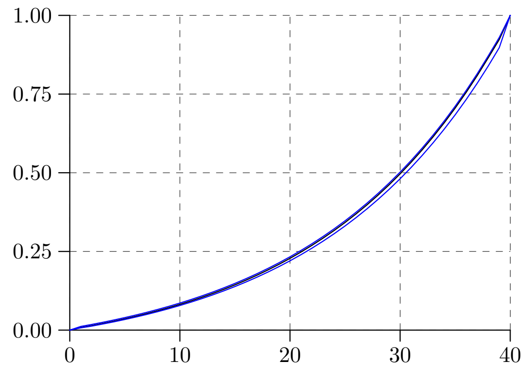


Figure 3: Upper line is the value of Player 1's best response to Player 2's risk-aware strategy, lower line is value of Player 2's best response Player 1's risk-aware strategy, middle line is critical the vector

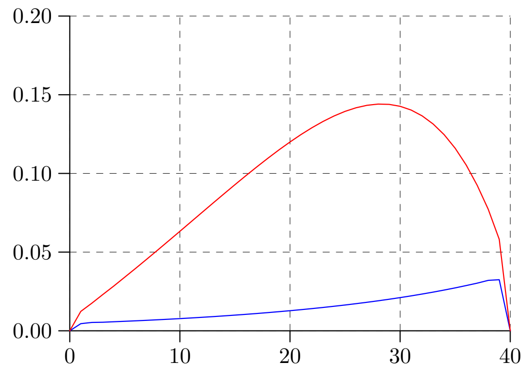


Figure 4: Upper line is the difference between the best response values against minimax. Lower line is the difference between the best response values against the risk-aware strategies.

where $\alpha = exp(-2\frac{\mu}{\sigma})$

Thus, if we know the typical outcome distribution of the game, we can estimate the riskiness as $\alpha = exp(-2\frac{\mu}{\sigma})$.

In 2007, the computer poker research group at the University of Alberta organized the First Man-Machine Poker Competition [10, p.79], where two professional poker players played against four different poker playing programs, collectively called Polaris. Out of the four sessions, the humans won two sessions, drew one, and lost one. The only program they lost to was named "Mr. Orange", and it was constructed by solving the game with respect to a modified utility function. The utility function was as follows:

$$u(v) = \begin{cases} v & \text{if } v \leq 0 \\ 1.07 \cdot v & \text{if } v > 0 \end{cases}$$

In other words, the program saw the game as if any winnings where 7% higher, while losses where left unmodified. A side effect of this choice was that the game was no longer a zero-sum game, since the 7% was not paid by the loser. The resulting program was very aggressive; according to Laak [10] (one of the human players), Mr. Orange "...was like a crazed, cocaine-driven maniac with an ax".

If we were to use the results of this paper to suggest an alternative utility function, we could use empirical observations about the game to estimate the riskiness of the game. Heads Up Limit Texas Hold'em has been observed [2] to have a standard deviation of 6.856 sb/hand. Currently, the best minimax algorithms produce poker playing programs that lose around 0.1 sb/hand against an optimal opponent [11]. We can then use the discussion in the previous section to figure out what riskiness balances out the 0.1 sb/hand suboptimality. Using the formula, we get the $\alpha = exp(-2 \cdot (-0.1)/6.856) \approx 1.03$, and the resulting modified utility function becomes

$$u'(v) = \frac{1.03^v - 1}{\ln 1.03}$$

Notice that $u(v)$ and $u'(v)$ are very close for typical values $v \in [-\sigma; \sigma]$. While the setting for the Man-Machine match was not exactly the same as our model, the similarity does provide some hope to the applicability of the approach.

9. EXTENSIONS

The most natural extension is to remove the requirement that outcomes must be integer. If we allow them to rational numbers instead, we can still do the same analysis. The explicit modelling of the game would require C/gcd game elements, where gcd is the greatest common divisor of all the outcomes of the game. This could be a very large number of game elements, but as our approach uses oblivious strategies, we are not hindered by this. The easiest way to prove performance guarantees for rational valued games is by scaling all number of the game (outcomes and amount of money) up with a constant, so that everything becomes integers. To do this, we need to know how scaling affects riskiness of a game.

PROPOSITION 15. *Multiplying all outcomes of a game A by constant k results in the riskiness becoming the k'th root of the previous riskiness:*

$$risk(k \cdot A) = risk(A)^{1/k}$$

PROOF. The property follows directly by the fact that exponentiating with the k'th root cancels out the scaling, except for a constant scaling of the whole utility function, and therefore $val((k \cdot A)^{\alpha^{1/k}} = k \cdot val(A^{\alpha}) = 0$. $\square$

Using this property, we can scale to integers, get the performance guarantee, and scale back to the original game. Doing this, we get the lower bound on winning probability for Player 1 to be

$$\frac{\alpha^{c_1} - 1}{\alpha^{C+max-gcd} - 1}$$

when the game starts from money division (c_1, c_2). In other words, the only change is that the -1 we got from the overlap for high index game elements is replaced with an expression that only depends on the input game. Notice that Algorithm 1 does not rely on the outcomes being integer, so it can be used directly on games with fractional outcomes; the scaling is only for the analysis.

Another interesting extension of the result is to the case where only one of the players has a restricted budget, while his opponent has infinite resources. The goal for the budget constrained player is to build his fortune, while avoiding going bankrupt in the process. Of course, for this to be interesting, the budget constrained player must have positive expectation; otherwise he will lose with probability 1. Assume wlog that it is Player 1 that is budget constrained, and that the value of the underlying game is positive. This implies that the game has low riskiness. We can now observe that in this case the performance guarantee given by Theorem 12 for a fixed c_1 converges to $1 - \alpha^{c_1}$ for $c_2 \to \infty$. Thus, the risk-aware strategies apply to one-sided budgets as well.

10. DISCUSSION AND FUTURE RESEARCH

An interesting observation on the riskiness estimate in section 8 is that the riskiness is closely tied to the ratio of mean over standard deviation. In portfolio management, this ratio is commonly called the Sharpe ratio [14] (with risk free rate 0). In many idealized settings, it is exactly the Sharpe ratio one wants to maximize. It does, however, have important shortcomings. Primarily, it relies on the outcomes being normally distributed, which is not in general the case for the scenarios we have examined in this paper. To the best of our knowledge, there is no known algorithm for maximizing the Sharpe ratio of a zero-sum game. However, we can observe that our proposed solution concept outperforms Sharpe ratio maximization on simple examples. To see why this is the case, examine the doubling game from the introduction. Both the doubled and the undoubled games have the same Sharpe ratio, so a slight perturbation would make a Sharpe ratio maximizer choose the wrong action. One can easily construct games where the scaling is with a larger constant to exacerbate the problem.

In this paper, we have only examined a single setup, where two players where playing under a budget. The general idea of deriving a utility function to use for solving a sub-problem leads to many open problems. For instance, in the Man-Machine poker tournament discussed in section 8, the true objective was not to bankrupt the opponent (you cannot; he has unlimited bankroll), but rather to have the most money after a fixed number of rounds has been played. This again leads to a non-linear utility function, but the exact function to be used in unclear.

11. REFERENCES

[1] R. J. Aumann and R. Serrano. An economic index of riskiness. *Journal of Political Economy*, 116:810–836, 2008.

[2] D. Billings. *Algorithms and Assessment in Computer Poker*. PhD thesis, University of Alberta, 2006.

[3] A. Condon. The complexity of stochastic games. *Information and Computation*, 96:203–224, 1992.

[4] A. Condon. On algorithms for simple stochastic games. *Advances in Computational Complexity Theory, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 13:51–73, 1993.

[5] L. de Alfaro, T. A. Henzinger, and O. Kupferman. Concurrent reachability games. *Theoretical Computer Science*, 386(3):188 – 217, 2007. Expressiveness in Concurrency.

[6] K. Etessami and M. Yannakakis. Recursive concurrent stochastic games. In *Proceedings of International Colloqium on Automata, Languages and Programming (ICALP)*, pages 324–335, 2006.

[7] H. Everett. Recursive games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games Vol. III*, volume 39 of *Annals of Mathematical Studies*. Princeton University Press, 1957.

[8] K. A. Hansen, R. Ibsen-Jensen, and P. B. Miltersen. The complexity of solving reachability games using value and strategy iteration. In *Proceedings of the 6th International Computer Science Symposium in Russia, CSR*, 2011.

[9] K. A. Hansen, M. Kouchý, N. Lauritzen, P. B. Miltersen, and E. P. Tsigaridas. Exact algorithms for solving stochastic games. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, 2011.

[10] M. Johanson. Robust strategies and counter-strategies: Building a champion level computer poker player. Master's thesis, University of Alberta, 2007.

[11] M. Johanson, M. Bowling, K. Waugh, and M. Zinkevich. Accelerating best response calculation in large extensive games. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, pages 258–265, 2011.

[12] H. Kuhn. A simplified two-person poker. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the theory of games I*, volume 24 of *Annals of Mathematical Studies*. Princeton University Press, 1950.

[13] P. B. Miltersen and T. B. Sørensen. A near-optimal strategy for a heads-up no-limit texas hold'em poker tournament. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, 2007.

[14] W. F. Sharpe. Mutual fund performance. *Journal of Business*, 1:119–138, 1966.

Efficient Nash Equilibrium Approximation through Monte Carlo Counterfactual Regret Minimization

Michael Johanson, Nolan Bard, Marc Lanctot, Richard Gibson, and Michael Bowling
University of Alberta
Edmonton, Alberta
{johanson,nolan,lanctot,rggibson,bowling}@cs.ualberta.ca

ABSTRACT

Recently, there has been considerable progress towards algorithms for approximating Nash equilibrium strategies in extensive games. One such algorithm, Counterfactual Regret Minimization (CFR), has proven to be effective in two-player zero-sum poker domains. While the basic algorithm is iterative and performs a full game traversal on each iteration, sampling based approaches are possible. For instance, chance-sampled CFR considers just a single chance outcome per traversal, resulting in faster but less precise iterations. While more iterations are required, chance-sampled CFR requires less time overall to converge. In this work, we present new sampling techniques that consider sets of chance outcomes during each traversal to produce slower, more accurate iterations. By sampling only the public chance outcomes seen by all players, we take advantage of the imperfect information structure of the game to (i) avoid recomputation of strategy probabilities, and (ii) achieve an algorithmic speed improvement, performing $O(n^2)$ work at terminal nodes in $O(n)$ time. We demonstrate that this new CFR update converges more quickly than chance-sampled CFR in the large domains of poker and Bluff.

Categories and Subject Descriptors

I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*Games*

General Terms

Algorithms

Keywords

Economic paradigms::Game theory (cooperative and non-cooperative)

1. INTRODUCTION

Extensive games are an intuitive formalism for modelling interactions between agents in a sequential decision making setting. One solution concept in such domains is a Nash equilibrium. In two-player zero-sum domains, this is equivalent to a minmax strategy, which minimizes each agent's expected worst-case performance. For games of moderate size, such a strategy can be found using linear programming [5]. For larger games, techniques such as Counterfactual Regret Minimization (CFR) [10] and the Excessive Gap Technique [3] require less memory than linear programming and are capable of finding an equilibrium in games (also known as **solving** a game) with up to 10^{12} game states.

CFR is an iterative procedure that resembles self-play. On each iteration, CFR performs a full game tree traversal and updates its entire strategy profile to minimize regret at each decision. Theoretical bounds suggest that the procedure takes a number of iterations at most quadratic in the size of a player's strategy [10, Theorem 4]. Thus, as we consider larger games, not only are more iterations required to converge, but each traversal becomes more time consuming. A variant known as **Chance-Sampled (CS)** CFR [6, 10] samples one set of chance outcomes per iteration and traverses only the corresponding portion of the game tree. Compared to the basic algorithm, this sampling procedure results in faster but less precise strategy updates. In large games, the drastic reduction in per-iteration time cost outweighs the increased number of iterations required for convergence to an optimal strategy.

While CS considers only a single set of chance outcomes per iteration, recent work [4] towards fast best-response computation has shown that tree traversal and evaluation can be accelerated by simultaneously considering sets of information sets for each player. This allows for the caching and reuse of computed values, and also allows a fast terminal node evaluation in which $O(n^2)$ work can often be done in $O(n)$ time. While best response calculation in large games was previously considered intractable, the new technique was shown to perform the computation in just over one day [4].

In this paper, we apply this new tree traversal to CFR, resulting in three new sampling variants: **Self-Public Chance Sampling (SPCS)**, **Opponent-Public Chance Sampling (OPCS)**, and **Public Chance Sampling (PCS)**. The new techniques reverse the previous trend in that they advocate less sampling: a small number of slow iterations, each updating a large number of information sets, yielding precise strategy updates while reusing computed values. In particular, PCS takes advantage of the computation reuse and fast terminal node evaluation used in accelerating the best response computation. We will prove the convergence of the new techniques, investigate their qualities, and demonstrate empirically that PCS converges more quickly to an equilibrium than CS in both poker and the game of Bluff.

2. BACKGROUND

An extensive game is a general model of sequential decision-making with imperfect information. Extensive games consist primarily of a game tree whose nodes correspond to **histories** (sequences) of actions $h \in H$. Each non-terminal history, h, has an associated **player** $P(h) \in N \cup \{c\}$ (where N is the set of players and c denotes **chance**) that selects an **action** $a \in A(h)$ at that history h. When $P(h) = c$, $f_c(a|h)$ is the (fixed) probability of

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

chance generating action a at h. We call h a **prefix** of history h', written $h \sqsubseteq h'$, if h' begins with the sequence h. Each **terminal history** $z \in Z \subset H$ has associated **utilities** for each player i, $u_i(z)$. In **imperfect information** games, histories are partitioned into **information sets** $I \in \mathcal{I}_i$ representing different game states that player i cannot distinguish between. For example, in poker, player i does not see the opponents' private cards, and thus all histories differing only in the private cards dealt to the opponents are in the same information set for player i. For histories $h, h' \in I$, the actions available at h and h' must be the same, and we denote this action set by $A(I)$. We also assume **perfect recall** that guarantees players always remember information that was revealed to them and the order in which it was revealed.

A **strategy for player i**, σ_i, is a function that maps each $I \in \mathcal{I}_i$ to a probability distribution over $A(I)$. We denote Σ_i as the set of all strategies for player i. A **strategy profile** is a vector of strategies $\sigma = (\sigma_1, \ldots, \sigma_{|N|})$, one for each player. We let σ_{-i} refer to the strategies in σ excluding σ_i.

Let $\pi^\sigma(h)$ be the probability of history h occurring if all players choose actions according to σ. We can decompose

$$\pi^\sigma(h) = \prod_{i \in N} \pi_i^\sigma(h) \prod_{\substack{h'a \sqsubseteq h \\ P(h')=c}} f_c(a|h')$$

into each player's and chance's contribution to this probability. Here, $\pi_i^\sigma(h)$ is the contribution from player i when playing according to σ_i. Let $\pi_{-i}^\sigma(h)$ be the product of all players' contribution (including chance) except that of player i. Furthermore, let $\pi^\sigma(h, h')$ be the probability of history h' occurring, given h has occurred with $\pi_i^\sigma(h, h')$, and $\pi_{-i}^\sigma(h, h')$ defined similarly.

Given a strategy profile, σ, we define a player's **best response** as a strategy that maximizes their expected payoff, assuming all other players play according to σ. The **best-response value** for player i is the value of that strategy, $b_i(\sigma_{-i}) = \max_{\sigma_i' \in \Sigma_i} u_i(\sigma_i', \sigma_{-i})$. A strategy profile σ is an **ϵ-Nash equilibrium** if no player can deviate from σ and gain more than ϵ; *i.e.* $u_i(\sigma) + \epsilon \geq \max_{\sigma_i' \in \Sigma_i} u_i(\sigma_i', \sigma_{-i})$ for all $i \in N$. If $\epsilon = 0$, then σ is a **Nash equilibrium** and every player is playing a best response.

In this paper, we will focus on **two-player zero-sum** games: $N = \{1, 2\}$ and $u_1(z) = -u_2(z)$ for all $z \in Z$. In this case, the **exploitability** of σ, $\epsilon_\sigma = (b_1(\sigma_2) + b_2(\sigma_1))/2$, measures how much σ loses to a worst case opponent when players alternate positions. A Nash equilibrium has an exploitability of 0.

Lastly, define $\mathcal{C} = \{h \in H : P(h) = c\}$ to be the set of all histories where it is chance's turn to act. We will assume that $\mathcal{C}$ can be partitioned into three sets with respect to player i: $\mathcal{S}_i$, $\mathcal{O}_i$, and $\mathcal{P}$. Each set contains the histories h whose actions $a \in A(h)$, or **chance events**, are observable only by player i ($\mathcal{S}_i$), only by player i's opponent ($\mathcal{O}_i$), or by both players ($\mathcal{P}$). We refer to chance events occurring at $h \in \mathcal{S}_i \cup \mathcal{O}_i$ as **private** and to chance events occurring at $h \in \mathcal{P}$ as **public**. In addition, we assume that the actions available to the players throughout the game are independent of the private chance events. These two assumptions hold for a large class of games, including poker as well as any Bayesian game with observable actions [8] (*e.g.*, Bluff or negotiation games); furthermore, games can often be modified by adding additional chance actions to satisfy the property.

2.1 Counterfactual Regret Minimization

Counterfactual Regret Minimization (CFR) resembles a self-play algorithm where we iteratively obtain strategy profiles σ^t based on regret values accumulated throughout previous trials. At each information set $I \in \mathcal{I}_i$, the expected value for player i at I under the current strategy is computed, assuming player i plays to reach I. This expectation is the **counterfactual value** for player i,

$$v_i(\sigma, I) = \sum_{z \in Z_I} u_i(z) \pi_{-i}^\sigma(z[I]) \pi^\sigma(z[I], z),$$

where Z_I is the set of terminal histories passing through I and $z[I]$ is the prefix of z contained in I. For each action $a \in A(I)$, these values determine the **counterfactual regrets** at iteration t, $r_i^t(I, a) = v_i(\sigma_{(I \to a)}^t, I) - v_i(\sigma^t, I)$, where $\sigma_{(I \to a)}$ is the profile σ except at I, action a is always taken. The regret $r_i^t(I, a)$ measures how much player i would rather play action a at I than play σ^t. The counterfactual regrets are accumulated and σ^t is updated by applying regret matching [2, 10] to the accumulated regrets. Regret matching is a regret minimizer; *i.e.*, over time, the average of the counterfactual regrets approaches 0. Minimizing counterfactual regret at each information set minimizes the **average overall regret** [10, Theorem 3], defined by

$$R_i^T = \max_{\sigma' \in \Sigma_i} \frac{1}{T} \sum_{t=1}^{T} \left(u_i(\sigma', \sigma_{-i}^t) - u_i(\sigma_i^t, \sigma_{-i}^t) \right).$$

It is well-known that in a two-player zero-sum game, minimizing average overall regret implies that the average profile $\overline{\sigma}^T$ is an approximate equilibrium. CFR produces an ϵ-Nash equilibrium in $O(|H||\mathcal{I}_i|/\epsilon^2)$ time [10, Theorem 4].

Rather than computing the exact counterfactual values on every iteration, one can instead sample the values using **Monte Carlo CFR (MCCFR)** [6]. **Chance-sampled (CS) CFR** [10] is an instance of MCCFR that considers just a single set of chance outcomes per iteration. In general, let $\mathcal{Q}$ be a set of subsets, or **blocks**, of the terminal histories Z such that the union of all blocks spans Z. For CS, $\mathcal{Q}$ is the partition of Z where two histories belong to the same block if and only if no two chance events differ. In addition, a probability distribution over $\mathcal{Q}$ is required and a block $Q \in \mathcal{Q}$ is sampled on each iteration, giving us the **sampled counterfactual value** for player i,

$$\tilde{v}_i(\sigma, I) = \sum_{z \in Z_I \cap Q} u_i(z) \pi_{-i}^\sigma(z[I]) \pi^\sigma(z[I], z)/q(z),$$

where $q(z)$ is the probability that z was sampled. In CS, we sample the blocks according to the likelihood of the chance events occurring, so that

$$q(z) = \prod_{\substack{ha \sqsubseteq z \\ h \in \mathcal{C}}} f_c(a|h).$$

The counterfactual regrets are then measured according to these sampled values, as opposed to "vanilla CFR" that uses the true values $v_i(\sigma, I)$. Sampling reduces enumeration to the smaller subset Q rather than all of Z, decreasing the amount of time required per iteration. For a fixed ϵ, CS requires more iterations than vanilla CFR to obtain an ϵ-Nash equilibrium; however, the overall computing time for CS is lower in poker games [9, Appendix A.5.2].

2.2 Accelerated Traversal and Evaluation

A recent paper describes how to accelerate the computation of the best response value in large extensive form games [4]. This technique traverses a game's **public tree**, which represents the state of the game visible to all players. The authors observe that each player's strategy must be independent of the other player's private information. As such, a player's action probabilities can be computed just once while considering the opponent's entire set of possible private states in one traversal.

In addition, the authors describe an efficient terminal node evaluation that considers a range of n information sets for each player in tandem. If the game's payoffs exhibit structure, then it may be possible to exploit this structure and reduce a naive $O(n^2)$ computation to $O(n)$. Examples of structured payoffs include games where utilities are affected by only certain factors within the players' information sets, such as in a negotiation game, and games where information sets can be ranked from weakest to strongest, such as in poker. This algorithmic speedup is not being used in any of the previously published equilibrium solvers. In Section 3, we describe how to use these ideas to produce a new equilibrium solver that outperforms the current state of the art.

2.3 Domains: Poker and Bluff

The Game of Poker. Our main poker game of interest is heads-up (*i.e.*, two-player) limit Texas hold'em poker, or simply **Texas hold'em**. The game uses a standard 52 card deck and consists of 4 betting rounds. In the first round, the **pre-flop**, each player is dealt two private cards. For subsequent rounds – in order, the **flop**, **turn**, and **river** – public community cards are revealed (3 at the flop and 1 at each of the turn and river). During each round, players sequentially take one of three actions: **fold** (forfeit the game), **call** (match the previous bet), or **raise** (increase the bet). There is a maximum of 4 raises per round, each with a fixed size, where the size is doubled on the final two rounds. If neither player folds, then the player with the highest ranked poker hand wins all of the bets.

Texas hold'em contains approximately 3.2×10^{14} information sets. The large size of the game makes an equilibrium computation intractable for all known algorithms; CFR would require more than ten petabytes of RAM and hundreds of CPU-years of computation. A common approach is to use state-space abstraction to produce a similar game of a tractable size by merging information sets or restricting the action space [1]. In Section 4, we consider several abstractions of Texas hold'em and two new variants of Texas hold'em that are small enough to compute equilibrium solutions using CFR without abstraction. The first new variant is **[2-1] hold'em**. The game is identical to Texas hold'em, except consists of only the first two betting rounds, the pre-flop and flop, and only one raise is allowed per round. This reduces the size of the game to 16 million information sets. Similarly, **[2-4] hold'em** has just two rounds, but the full four raises are allowed per round, resulting in 94 million information sets in total. In both [2-1] hold'em and [2-4] hold'em, the size of a raise doubles from the pre-flop to the flop.

The Game of Bluff. Bluff, also known as Liar's Dice, Dudo, and Perudo, is a dice-bidding game. In our version, Bluff(D_1,D_2), each die has six sides with faces 1 to 5 and a star: $\star$. Each player i rolls D_i of these dice and looks at them without showing them to their opponent. On each round, players alternate by bidding on the outcome of all dice in play until one player claims that the other is bluffing (*i.e.*, claims that the bid does not hold). A bid consists of a **quantity** of dice and a **face** value. A face of $\star$ is considered "wild" and counts as matching any other face. For example, the bid 2-5 represents the claim that there are at least two dice with a face of 5 or $\star$ among both players' dice. To place a new bid, the player must increase either the quantity or face value of the current bid; in addition, lowering the face is allowed if the quantity is increased. The player calling bluff wins the round if the opponent's last bid is incorrect, and loses otherwise. The losing player removes one of their dice from the game and a new round begins, starting with the player who won the previous round. When a player has no more dice left, they have lost the game. A utility of $+1$ is given for a win and -1 for a loss.

In this paper, we restrict ourselves to the case where $D_1 = D_2 = 2$, a game containing 352 million information sets. Note that since Bluff(2,2) is a multi-round game, the expected values of Bluff(1,1) are precomputed for payoffs at the leaves of Bluff(2,1), which is then solved for leaf payoffs in the full Bluff(2,2) game.

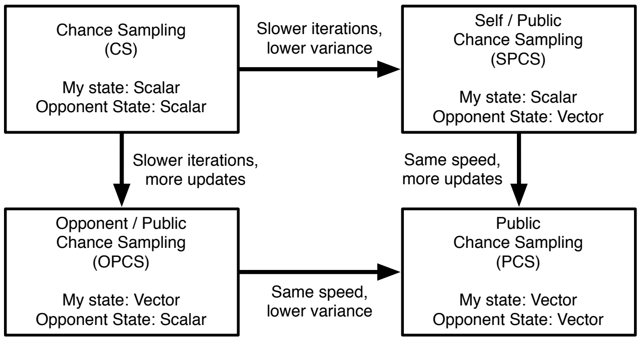


Figure 1: Relationship between MCCFR variants

3. NEW MONTE CARLO CFR VARIANTS

Before presenting our new CFR update rules, we will begin by providing a more practical description of chance-sampled CFR. On each iteration, we start by sampling all of chance's actions: the public chance events visible to each player, as well as the private chance events that are visible to only a subset of the players. In poker, this corresponds to randomly choosing the public cards revealed to the players, and the private cards that each player is dealt. In the game of Bluff, there are no public chance events, and only private chance events are sampled for each player. Next, we recursively traverse the portion of the game tree that is reachable given the sampled chance events, and explore all of the players' actions. On the way from the root to the leaves, we pass forward two scalar values: the probability that each player would take actions to reach their respective information sets, given their current strategy and their private information. On the way back from the leaves to the root, we return a single scalar value: the sampled counterfactual value $\tilde{v}_i(\sigma, I)$ for player i. At each choice node for player i, these values are all that is needed to calculate the regret for each action and update the strategy. Note that at a terminal node $z \in Z$, it takes $O(1)$ work to determine the utility for player i, $u_i(z)$.

We will now describe three different methods of sampling chance events that have slower iterations, but do more work on each iteration. Figure 1 shows the relationship between CS and these three new variants, all of which belong to the MCCFR family [6] of update rules.

Opponent-Public Chance Sampling. Consider a variation on CS, where instead of sampling at every chance node, we sample an action for just the opponent's chance and the public chance events while enumerating all of the possible outcomes at our private chance events. We will call this variant Opponent-Public Chance Sampling (OPCS). This can be formalized within the MCCFR framework by letting $\mathcal{Q}$ be the partition of Z such that two histories fall into the same block if and only if the actions taken at opponent and public chance events match. The probability that z is sampled is then

$$q(z) = \prod_{\substack{ha \sqsubseteq z \\ h \in \mathcal{O}_{-i} \cup \mathcal{P}}} f_c(a|h).$$

Naively, we could use the same recursive tree walk that we used for CS to perform this update, by doing one tree walk for each of our private chance outcomes in turn. However, this update allows us to traverse the sampled portion of the game tree in a much more efficient way. Since our opponent does not observe our private chance events, their strategy and choice of actions, given their single sampled chance event, cannot depend on which information set we are in. This means that we can update all of our information sets that are consistent with the current game state and the sampled public chance events at the same time, thus amortizing the cost of walking the tree over many updates. This can be achieved by a new recursive tree walk that passes forwards a vector for us (our probability of reaching the current game state with each of our private chance outcomes) and a scalar for the opponent (their probability of reaching the current game state with their single sampled private chance outcome), and returns a vector of values (our counterfactual value for each of our private chance outcomes).

At terminal nodes, we must evaluate n possible game states, each consisting of a different private chance outcome for us and one chance outcome for the opponent. This requires $O(n)$ time. In comparison to CS, each iteration of OPCS is slower, but performs more work by updating a much larger number of information sets.

Self-Public Chance Sampling. In OPCS, we enumerate over all of our possible private chance outcomes. Alternatively, we can instead enumerate over all of our opponent's private chance outcomes while sampling our own private chance outcomes and the public chance outcomes. We will call this variant Self-Public Chance Sampling (SPCS). This can similarly be formalized by defining $\mathcal{Q}$ to be the partition of Z that separates histories into different blocks whenever the actions taken at our private or public chance events differ, where

$$q(z) = \prod_{\substack{ha \sqsubseteq z \\ h \in \mathcal{S}_i \cup \mathcal{P}}} f_c(a|h)$$

is the probability of sampling terminal history z.

As in OPCS, we can use an efficient recursive tree walk to perform this update. Since we cannot observe the opponent's private chance events, our strategy and choice of actions cannot depend on which information set they are in. Thus, when computing our counterfactual value, we will consider every possible private chance outcome for our opponent. Doing so forms a more accurate estimate of the true counterfactual value for our sampled outcome, compared to the noisy estimate CS and OPCS obtain through one sampled opponent private chance outcome. The SPCS tree walk passes forward a scalar for ourselves (the probability of reaching the current game state with our single chance outcome) and a vector for the opponent (their probabilities of reaching the current game state with each of their private chance outcomes), and returns a scalar (the counterfactual value for our sampled outcome).

At terminal nodes, we must evaluate up to n possible game states, formed by our single chance outcome and up to n possible chance outcomes for the opponent. This requires $O(n)$ time. In comparison to CS, each iteration is slower and performs the same number of updates to the strategy, but each update is based off of much more precise estimates.

Public Chance Sampling. We will now introduce the core contribution of this work, called Public Chance Sampling (PCS), that combines the advantages of both of the previous two updates, while taking advantage of efficient terminal node evaluation to keep the time cost per iteration in $O(n)$. In PCS, we sample only the public chance events, and consider all possible private chance events for ourself and for the opponent. In other words, we define $\mathcal{Q}$ to be the partition of Z that separates histories into different blocks whenever the actions taken at a public chance event differ, where

$$q(z) = \prod_{\substack{ha \sqsubseteq z \\ h \in \mathcal{P}}} f_c(a|h)$$

is the probability of sampling $z \in Z$.

PCS relies on the property that neither us nor our opponent can observe the other's private chance events, and so the action probabilities for each remain the same across the other's private information. Thus, we can perform a CFR update through a recursive tree walk with the following structure. On the way from the root to the leaves, we will pass forwards two vectors: one containing the probabilities of us and one containing the probabilities of the opponent reaching the current game state, for each player's n possible private chance outcomes. On the way back, we will return a vector containing the counterfactual value for each of our n information sets.

At the terminal nodes, we seemingly have an $O(n^2)$ computation, as for each of our n information sets, we must consider all n of the opponent's possible private outcomes in order to compute our utility for that information set. However, if the payoffs at terminal nodes are structured in some way, we can often reduce this to an $O(n)$ evaluation that returns exactly the same value as the $O(n^2)$ evaluation [4]. Doing so gives PCS the advantage of both SPCS (accurate strategy updates) and OPCS (many strategy updates) for the same evaluation cost of either.

3.1 Algorithm

The three new chance-sampling variants, along with CS, are shown in Algorithm 1. The WalkTree function traverses down the game tree by recursively concatenating actions, starting with the empty history $h = \emptyset$, and updates player i's regrets and average strategy on the way back up. Two vectors are maintained, one for player i, $\vec{\pi}_i$, and one for the opponent, $\vec{\pi}_{-i}$. These vectors keep track of the probabilities of reaching each information set consistent with the current history h, with each element corresponding to a different private chance outcome for that player. In CS, both vectors have length one (*i.e.*, are scalars). In OPCS, $\vec{\pi}_{-i}$ has length one because the opponent's private chance events are being sampled. Similarly, in SPCS, $\vec{\pi}_i$ has length one.

When the current sequence h is a terminal history (line 6), the utility is computed and returned. At line 7, $\vec{f}_{c,i}(h)$ and $\vec{f}_{c,-i}(h)$ are the vectors corresponding to the probability distribution over player i's and the opponent's private chance outcomes, respectively, and $\odot$ represents element-wise vector multiplication. Again, one or both vectors may have length one depending on the selected variant, in which case the single element is always 1. For OPCS and PCS, $\vec{u}_i$ is a vector containing a utility for each of player i's private outcomes; for SPCS and CS, $\vec{u}_i$ is a length one vector corresponding to the utility for player i's sampled private outcome. PCS uses the $O(n^2)$ to $O(n)$ algorithmic improvement to compute $\vec{u}_i$, which will be described in Section 3.2.

Chance events are handled by lines 12 to 18. When one of the four conditions at line 12 holds, we are at a chance event that is to be sampled; otherwise, we consider all possible chance events at h. In the latter case, we must take a dummy action (line 16) simply to continue traversing the tree. This action has no effect on the remainder of the tree walk due to our assumption that player actions are independent of private chance events.

Lines 19 to 42 handle the cases where h is a decision node for one of the players. First, lookupInfosets(h) retrieves all of the information sets consistent with h and the current player $P(h)$'s range of possible private outcomes, whether sampled ($|\vec{I}| = 1$) or not. Next, at line 21, regret matching [2, 10] determines the current strategy $\vec{\sigma}$, a vector of action probabilities for each retrieved information set (and thus, in general, a vector of vectors). Regret matching assigns action probabilities according to

$$\sigma[a][I] = \begin{cases} r_I^+[a] / \sum_{b \in A(I)} r_I^+[b] & \text{if } \sum_{b \in A(I)} r_I^+[b] > 0 \\ 1/|A(I)| & \text{otherwise,} \end{cases}$$

Algorithm 1 PCS Algorithm

```
1: Require: a variant $v \in \{\text{CS}, \text{OPCS}, \text{SPCS}, \text{PCS}\}$.
2: Initialize regret tables: $\forall I, r_I[a] \leftarrow 0$.
3: Initialize cumulative strategy tables: $\forall I, s_I[a] \leftarrow 0$.
4:
5: function WalkTree($h$, $i$, $\vec{\pi}_i$, $\vec{\pi}_{-i}$):
6:     if $h \in Z$
7:         return $\vec{f}_{c,i}(h) \odot \vec{u}_i\left(h \mid \vec{\pi}_{-i} \odot \vec{f}_{c,-i}(h)\right)$
8:     end if
9:     if    ($v$ = PCS and $h \in \mathcal{P}$)
10:        or ($v$ = SPCS and $h \in \mathcal{S}_i \cup \mathcal{P}$)
11:        or ($v$ = OPCS and $h \in \mathcal{O}_i \cup \mathcal{P}$)
12:        or ($v$ = CS and $h \in \mathcal{C}$)
13:         Sample outcome $a \in A(h)$ with probability $f_c(a|h)$
14:         return WalkTree($ha$, $i$, $\vec{\pi}_i$, $\vec{\pi}_{-i}$)
15:     else if $h \in \mathcal{C}$
16:         Select dummy outcome $a \in A(h)$
17:         return WalkTree($ha$, $i$, $\vec{\pi}_i$, $\vec{\pi}_{-i}$)
18:     end if
19:     $\vec{I} \leftarrow$ lookupInfosets($h$)
20:     $\vec{u} \leftarrow \vec{0}$
21:     $\vec{\sigma} \leftarrow$ regretMatching($\vec{I}$)
22:     for each action $a \in A(h)$ do
23:         if $P(h) = i$
24:             $\vec{\pi}_i' \leftarrow \vec{\sigma}[a] \odot \vec{\pi}_i$
25:             $\vec{u}' \leftarrow$ WalkTree($ha$, $i$, $\vec{\pi}_i'$, $\vec{\pi}_{-i}$)
26:             $\vec{m}[a] \leftarrow \vec{u}'$
27:             $\vec{u} \leftarrow \vec{u} + \vec{\sigma}[a] \odot \vec{u}'$
28:         else
29:             $\vec{\pi}_{-i}' \leftarrow \vec{\sigma}[a] \odot \vec{\pi}_{-i}$
30:             $\vec{u}' \leftarrow$ WalkTree($ha$, $i$, $\vec{\pi}_i$, $\vec{\pi}_{-i}'$)
31:             $\vec{u} \leftarrow \vec{u} + \vec{u}'$
32:         end if
33:     end for
34:     if $P(h) = i$
35:         for $I \in \vec{I}$ do
36:             for $a \in A(I)$ do
37:                 $r_I[a] \leftarrow r_I[a] + m[a][I] - u[I]$
38:                 $s_I[a] \leftarrow s_I[a] + \pi_i[I]\sigma[a][I]$
39:             end for
40:         end for
41:     end if
42:     return $\vec{u}$
43:
44: function Solve():
45:     for $t \in \{1, 2, 3, \cdots\}$ do
46:         for $i \in N$ do
47:             WalkTree($\emptyset$, $i$, $\vec{1}$, $\vec{1}$)
48:         end for
49:     end for
```

where $r_I^+[a] = \max\{r_I[a], 0\}$. We then iterate over each action $a \in A(h)$, recursively obtaining the expected utilities for a at each information set (line 25 or 30). When $P(h) = i$, these utilities are stored (line 26) and used to update the regret at each information set (line 37), while the current strategy $\vec{\sigma}$ weights both the returned expected utility at h (line 27) and the average strategy update (line 38). Note that at line 31, we do not weight $\vec{u}'$ by $\vec{\sigma}[a]$ since the opponent's reaching probabilities are already factored into the utility computation (line 7).

After iterating over the outer loop of Solve() (line 45) for many iterations, an ϵ-Nash equilibrium is obtained from the accumulated strategies: $\bar{\sigma}(I, a) = s_I[a] / \sum_{b \in A(I)} s_I[b]$.

3.2 Efficient Terminal Node Evaluation

We now describe how PCS computes a vector of expected utilities $\vec{u}_i(h \mid \vec{\pi}_{-i})$ at line 7 for player i's n private outcomes in $O(n)$ time. As we have already noted, Johanson *et al.* [4] gave a detailed description for how to do this in poker. In this section, we will describe an efficient terminal node evaluation for Bluff(D_1, D_2).

Every game ends with one player calling bluff, and the payoffs ($+1$ or -1) are determined solely by whether or not the last bid holds. Let x-y be the last such bid. We now must discriminate between cases where there are less than and where there are at least x dice showing face y or $\star$.

At the terminal history h, we have a vector of reach probabilities $\vec{\pi}_{-i}$ for each of the opponent's n possible dice rolls. Let $\vec{X}_{-i}$ be a vector of length $D_{-i} + 1$, where the element $X_{-i}[j]$ ($0 \leq j \leq D_{-i}$) equals the probability of the opponent reaching h with exactly j dice showing face y or $\star$. $\vec{X}_{-i}$ is constructed in $O(n)$ time by iterating over each element of $\vec{\pi}_{-i}$, adding the probability to the appropriate entry of $\vec{X}_{-i}$ at each step. We can then compute the expected utility for player i with exactly j of his or her dice showing face y or $\star$. If player i called bluff, this expected utility is

$$U_i[j] = \sum_{\ell=0}^{x-j-1} (+1) \cdot X_{-i}[\ell] + \sum_{\ell=x-j}^{D_{-i}} (-1) \cdot X_{-i}[\ell];$$

if the opponent called bluff, the expected utility is $-U_i[j]$. Constructing $\vec{U}_i$ takes $O(n)$ time. Finally, we iterate over all $k \in \{1, ..., n\}$ and set $u_i[k] = U_i[x_k]$, where x_k is the number of dice showing face y or $\star$ in player i's kth private outcome. In total, the process takes $3O(n) = O(n)$ time.

3.3 Theoretical Analysis

CS, OPCS, SPCS, and PCS all belong to the MCCFR family of algorithms. As such, we can apply the general results for MCCFR to obtain a probabilistic bound on the average overall regret for CS and our new algorithms. Recall that in a two-player zero-sum game, minimizing average overall regret produces an ϵ-Nash equilibrium. The proof of Theorem 1 is in the appendix.

THEOREM 1. *For any $p \in (0, 1]$, when using CS, OPCS, SPCS, or PCS, with probability at least $1 - p$, the average overall regret for player i is bounded by*

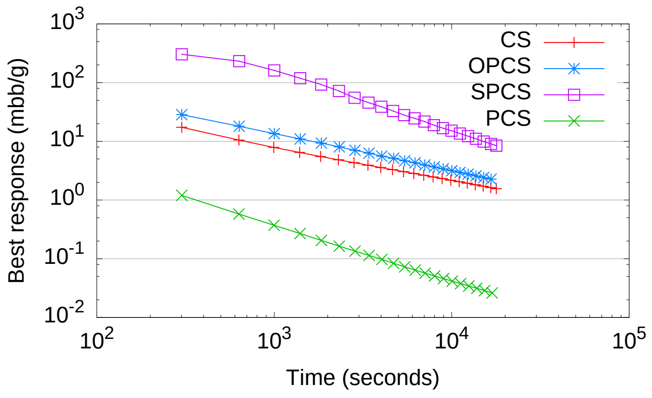
$$R_i^T \leq \left(1 + \frac{2}{\sqrt{p}}\right) \frac{\Delta_{u,i} M_i \sqrt{A_i}}{\sqrt{T}},$$

where M_i is a property of the game satisfying $\sqrt{|\mathcal{I}_i|} \leq M_i \leq |\mathcal{I}_i|$, $\Delta_{u,i} = max_{z,z'} |u_i(z) - u_i(z')|$, and $A_i = \max_{I \in \mathcal{I}_i} |A(I)|$.

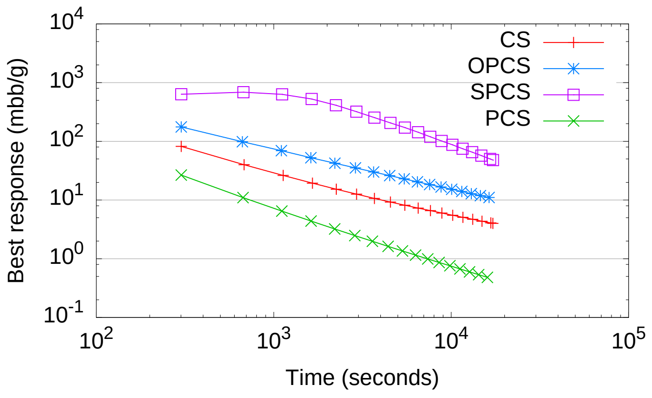
4. RESULTS

The efficacy of these new updates are examined through an empirical analysis in both poker and Bluff. We begin the analysis by examining the performance of CS, SPCS, OPCS and PCS in two small games, [2-1] hold'em and [2-4] hold'em. We will then present the performance of CS and PCS in a set of Texas hold'em abstract games, to investigate their usefulness under the conditions of the Annual Computer Poker Competition. Finally, we will apply CS and PCS to the Bluff domain.

Poker. [2-1] hold'em and [2-4] hold'em are games that are small enough to be tractably solved using all four of the CFR variants we are investigating: CS, SPCS, OPCS and PCS. As discussed in Section 3, SPCS, OPCS and PCS all perform $O(n)$ work at each



(a) $[2-1]$ hold'em, 16 million information sets



(b) $[2-4]$ hold'em, 94 million information sets

Figure 2: Log-log graphs displaying convergence of best response values over time for different CFR update methods in two small unabstracted hold'em like poker games. Best response values are in milli-big-blinds per game (mbb/g). Each curve shows the average performance over five independent runs.

terminal state, and are thus of comparable speed. However, all three require more time per iteration than CS, and to converge faster than CS, the advantage of each approach (more precise updates, more work per iteration, or both) must overcome this speed penalty.

Figure 2 shows the convergence of CS, OPCS, SPCS and PCS towards an optimal strategy in these small hold'em variants. We see that SPCS and OPCS converge slower than CS; the difference in speed is too great for the higher quality iterations. However, we find that PCS converges much more quickly than CS in these small games.

While [2-1] hold'em and [2-4] hold'em can be tractably solved using CFR, solving the much larger game of Texas hold'em is intractable. A common procedure used by competitors in the Annual Computer Poker Competition is to use a state-space abstraction technique to produce a smaller, similar game that can be tractably solved, and the resulting **abstract strategy** can then be used to select actions in the original game. The abstract strategy is an ϵ-Nash equilibrium in the abstract game, and we can measure its rate of convergence by calculating a best response within the abstract game. A critical choice in this procedure is the granularity of the abstraction. In practice, larger and finer-grained abstractions take longer to solve, but result in better approximations to a Nash equilibrium [4].

In Figure 3, we apply the CS and PCS algorithms to four sizes of abstract Texas hold'em games. The abstraction technique used in each is Percentile $E[HS^2]$, as described in [10], which merges information sets together if the chance events assign similar strength to a player's hand. An n-bucket abstraction branches the chance outcomes into n categories on each round.

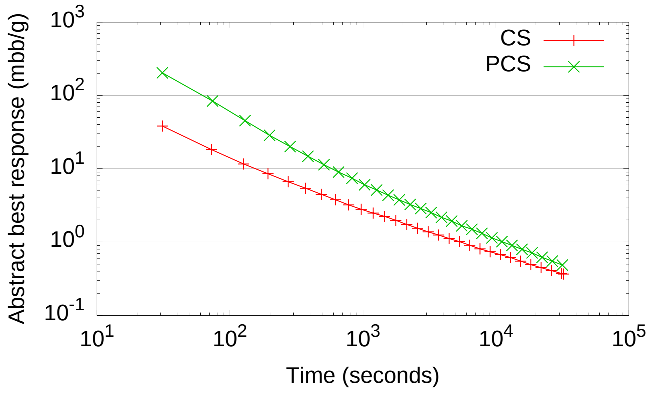
In the smallest abstract game in Figure 3a, we find that CS converges more quickly than PCS. As we increase the abstraction granularity through Figures 3b, 3c and 3d, however, we find that PCS matches and then surpasses CS in the rate of convergence. In each of these games, the chance sampling component samples outcomes in the real game and then maps this outcome to its abstract game equivalent. When a small abstraction is used, this means that many of the information sets being updated by PCS in one iteration will share the same bucket, and some of the benefit of updating many information sets at once is lost. In larger abstract games, this effect is diminished and PCS is of more use.

In the Annual Computer Poker Competition, many competitors submit entries that are the result of running CFR on very large abstract games. Computing a best response within such abstractions, as we did in Figure 3, is often infeasible (as many competitors use abstractions with imperfect recall). In these circumstances, we can instead evaluate a strategy based on its performance in actual games against a fixed opponent. We can use this approach to evaluate the strategies generated by CS and PCS at each time step, to investigate how PCS and CS compare in very large games.[1]

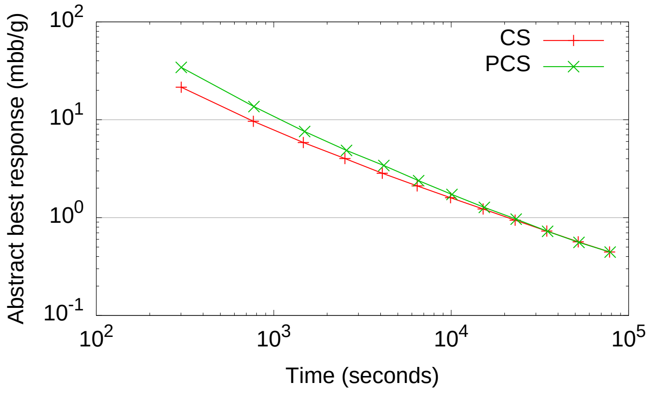
The results of this experiment are presented in Figure 4. The opponent in each match is Hyperborean 2010.IRO, which took third place in the 2010 Annual Computer Poker Competition's heads-up limit Texas hold'em instant runoff event. The y-axis shows the average performance in milli-big-blinds per game (mbb/g) over a 10-million hand match of duplicate poker, and the results are accurate to ± 1 mbb/g (so the difference in curves is statistically significant). The abstraction used for CS and PCS in this experiment uses imperfect recall and has 880 million information sets, and is similar to but slightly larger than Hyperborean's abstraction, which contains 798 million information sets. At each time step, the strategies produced by PCS perform better against Hyperborean than those produced by CS. Consider the horizontal difference between points on the curves, as this indicates the additional amount of time CS requires to achieve the same performance as PCS. As the competition's winner is decided based on one-on-one performance, this result suggests that PCS is an effective choice for creating competition strategies.

Bluff. Bluff(2,2) is small enough that no abstraction is required. Unlike poker, all of the dice rolls are private and there are no public chance events. In this domain, one iteration of PCS is equivalent to a full iteration of vanilla CFR (*i.e.*, no sampling). However, the reordering of the computation and the fast terminal node evaluation allows PCS to perform the iteration more efficiently than vanilla CFR. Figure 5 shows the convergence rates of CS and PCS in Bluff on a log-log scale. We notice that PCS converges towards equilibrium significantly faster than CS does. As noted earlier, PCS has two speed advantages: the fast terminal node evaluation, and the ability to reuse the opponent's probabilities of reaching an information set for many of our own updates. By comparison, vanilla

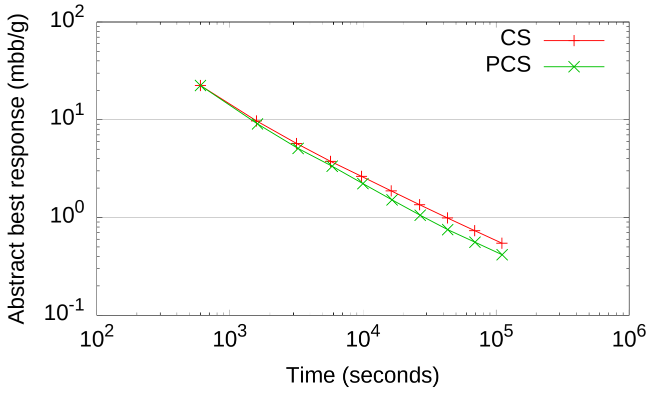
[1]Another possible evaluation metric is to compute the real-game exploitability of the strategies. However, the overfitting effect described in [4] makes the results unclear, as a strategy can become more exploitable in the real game as it approaches an equilibrium in the abstract game.



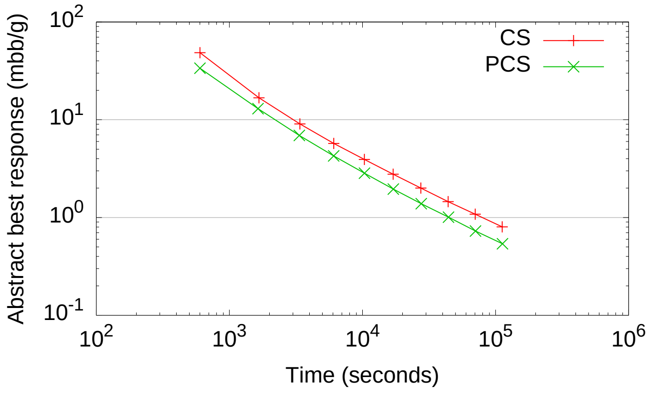
(a) 5 buckets, 3.6 million information sets



(b) 8 buckets, 23.6 million information sets



(c) 10 buckets, 57.3 million information sets



(d) 12 buckets, 118.6 million information sets

Figure 3: Log-log graphs displaying convergence of abstract best response values over time for different CFR update methods in two perfect recall abstractions of heads-up limit Texas hold'em poker. Best response values are in milli-big-blinds per game (mbb/g). Each curve shows the average performance over five independent runs.

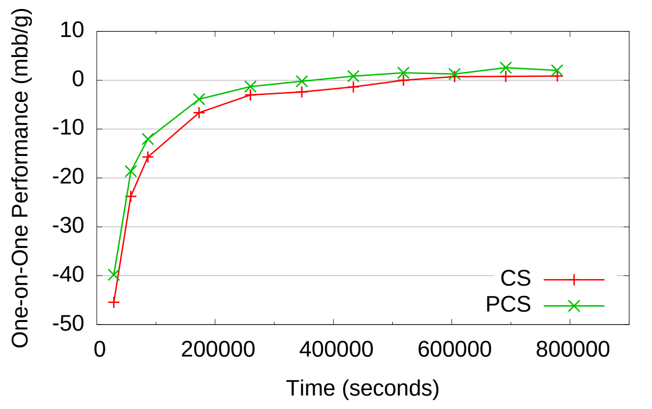


Figure 4: Performance of CS and PCS strategies in a large abstraction against a fixed, strong opponent.

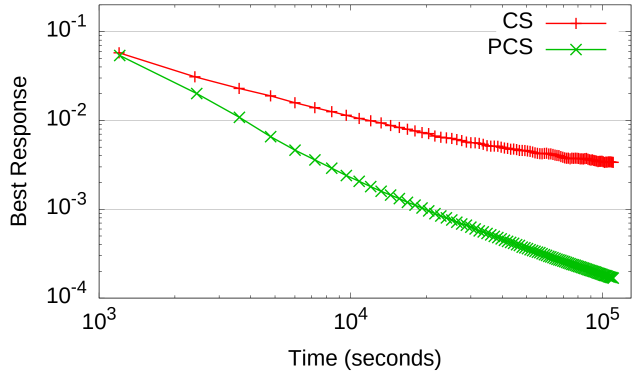


Figure 5: Log-log graph showing convergence of CS and PCS towards an equilibrium in Bluff(2,2). Each curve shows the average performance over five independent runs.

CFR would traverse the action space 441 times to do the work of 1 PCS traversal. Similar to Figure 4 in the poker experiments, we can also compare the performance of CS and PCS strategies against a fixed opponent: an ϵ-Nash equilibrium for Bluff(2,2). This experiment is presented in Figure 6, and the fixed opponent is the final data point of the PCS line; the results are similar if the final CS data point is used. This result shows that PCS is also more efficient than CS at producing effective strategies for one-on-one matches.

5. CONCLUSION

Chance Sampled CFR is a state-of-the-art iterative algorithm for approximating Nash equilibria in extensive form games. In this work, we presented three new CFR variants that perform less sampling than the standard approach. They perform slower but more efficient and precise iterations. We empirically demonstrated that Public Chance Sampling converges faster than Chance Sampling on

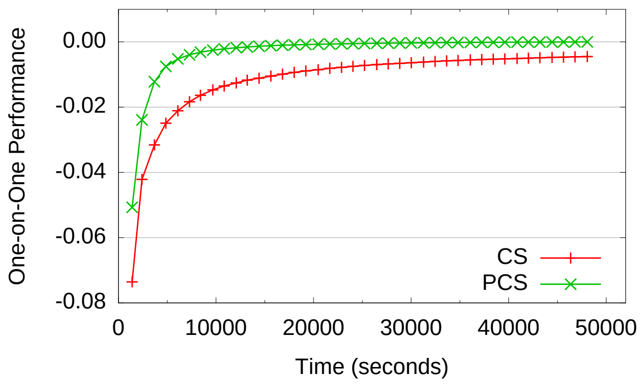


Figure 6: Performance of CS and PCS strategies against an ϵ-Nash equilibrium in Bluff(2,2)

large games, resulting in a more efficient equilibrium approximation algorithm demonstrated across multiple domains. Future work will look to tighten the theoretical bounds on the new algorithms to prove that they can outperform Chance Sampling.

Acknowledgments

The authors would like to thank the members of the Computer Poker Research Group at the University of Alberta for helpful conversations pertaining to this research. This research was supported by NSERC, Alberta Innovates Technology Futures, and the use of computing resources provided by WestGrid and Compute Canada.

6. REFERENCES

[1] A. Gilpin, T. Sandholm, and T. B. Sørensen. Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of texas hold'em poker. In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2007.

[2] S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

[3] S. Hoda, A. Gilpin, J. Peña, and T. Sandholm. Smoothing techniques for computing nash equilibria of sequential games. *Mathematics of Operations Research*, 35(2):494–512, 2010.

[4] M. Johanson, K. Waugh, M. Bowling, and M. Zinkevich. Accelerating best response calculation in large extensive games. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

[5] D. Koller, N. Megiddo, and B. von Stengel. Fast algorithms for finding randomized strategies in game trees. In *Annual ACM Symposium on Theory of Computing, STOC'94*, pages 750–759, 1994.

[6] M. Lanctot, K. Waugh, M. Zinkevich, and M. Bowling. Monte Carlo sampling for regret minimization in extensive games. In *Advances in Neural Information Processing Systems 22 (NIPS)*, 2009.

[7] M. Lanctot, K. Waugh, M. Zinkevich, and M. Bowling. Monte Carlo sampling for regret minimization in extensive games. Technical Report TR09-15, University of Alberta, 2009.

[8] M. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press, 1994.

[9] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione. Regret minimization in games with incomplete information. Technical Report TR07-14, Department of Computing Science, University of Alberta, 2007.

[10] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems 20 (NIPS)*, 2008.

APPENDIX

Proof of Theorem 1. Let $\vec{a}_i$ be a subsequence of a history such that it contains only player i's actions in that history, and let $\vec{A}_i$ be the set of all such subsequences. Let $\mathcal{I}_i(\vec{a}_i)$ be the set of all information sets where player i's action sequence up to that information set is $\vec{a}_i$. Without loss of generality, assume $i = 1$. Let $\mathcal{D} = \mathcal{C}, \mathcal{O}_1 \cup \mathcal{P}, \mathcal{S}_1 \cup \mathcal{P}$, or $\mathcal{P}$ depending on whether we are using CS, OPCS, SPCS, or PCS respectively. The probability of sampling terminal history z is then

$$q(z) = \prod_{\substack{ha \sqsubseteq z \\ h \in \mathcal{D}}} f_c(a|h). \tag{1}$$

Let $\vec{a}_i \in \vec{A}_i$, $B = \mathcal{I}_i(\vec{a}_i)$, and let $Q \in \mathcal{Q}$. By [7, Theorem 7], it suffices to show that

$$Y = \sum_{I \in B} \left(\sum_{z \in Z_I \cap Q} \pi^\sigma_{-1}(z[I]) \pi^\sigma(z[I], z)/q(z) \right)^2 \leq 1.$$

By (1) and definition of π^σ_{-i}, we have

$$Y = \sum_{I \in B} \left(\sum_{z \in Z_I \cap Q} \pi^\sigma_2(z[I]) \pi^\sigma_{1,2}(z[I], z) \prod_{\substack{ha \sqsubseteq z \\ h \in \mathcal{C} \setminus \mathcal{D}}} f_c(a|h) \right)^2 . \tag{2}$$

Now by the definition of $\mathcal{Q}$, for each $h \in \mathcal{D}$, there exists a unique $a^*_h \in A(h)$ such that if $z \in Q$ and $h \sqsubseteq z$, then $ha^*_h \sqsubseteq z$. Next, we define a new probability distribution on chance events according to

$$\hat{f}_c(a|h) = \begin{cases} 1 & \text{if } h \in \mathcal{D}, a = a^*_h \\ 0 & \text{if } h \in \mathcal{D}, a \neq a^*_h \\ f_c(a|h) & \text{if } h \in \mathcal{C} \setminus \mathcal{D}. \end{cases}$$

Notice that $\prod_{ha \sqsubseteq z, h \in \mathcal{D}} \hat{f}_c(a|h)$ is 1 if $z \in Q$ and is 0 if $z \notin Q$. Thus from (2), we have

$$\begin{aligned} Y &= \sum_{I \in B} \left(\sum_{z \in Z_I} \pi^\sigma_2(z[I]) \pi^\sigma_{1,2}(z[I], z) \prod_{\substack{ha \sqsubseteq z \\ h \in \mathcal{C}}} \hat{f}_c(a|h) \right)^2 \\ &= \sum_{I \in B} \left(\sum_{z \in Z_I} \hat{\pi}^\sigma_{-1}(z[I]) \hat{\pi}^\sigma(z[I], z) \right)^2 \\ &\quad \text{where } \hat{\pi}^\sigma \text{ is } \pi^\sigma \text{ except } f_c \text{ is replaced by } \hat{f}_c \\ &= \sum_{I \in B} \left(\sum_{h \in I} \hat{\pi}^\sigma_{-1}(h) \sum_{z \in Z_I} \hat{\pi}^\sigma(h, z) \right)^2 \\ &= \sum_{I \in B} \left(\sum_{h \in I} \hat{\pi}^\sigma_{-1}(h) \right)^2 \leq 1, \end{aligned}$$

where the last inequality follows by [7, Lemma 16]. $\square$.

Session 3E
Game Theory III

Computing Optimal Strategy against Quantal Response in Security Games

Rong Yang[+], Fernando Ordonez[+,*], Milind Tambe[+]
[+] University of Southern California, Los Angeles, CA, US
[*] University of Chile, Santiago, Chile
[+]{yangrong,fordon,tambe}@usc.edu
[*]fordon@dii.uchile.cl

ABSTRACT

To step beyond the first-generation deployments of attacker-defender security games – for LAX Police, US FAMS and others – it is critical that we relax the assumption of perfect rationality of the human adversary. Indeed, this assumption is a well-accepted limitation of classical game theory and modeling human adversaries' bounded rationality is critical. To this end, quantal response (QR) has provided very promising results to model human bounded rationality. However, in computing optimal defender strategies in real-world security games against a QR model of attackers, we face difficulties including (1) solving a nonlinear non-convex optimization problem efficiently for massive real-world security games; and (2) addressing constraints on assigning security resources, which adds to the complexity of computing the optimal defender strategy.

This paper presents two new algorithms to address these difficulties: GOSAQ can compute the globally optimal defender strategy against a QR model of attackers when there are no resource constraints and gives an efficient heuristic otherwise; PASAQ in turn provides an efficient approximation of the optimal defender strategy with or without resource constraints. These two novel algorithms are based on three key ideas: (i) use of a binary search method to solve the fractional optimization problem efficiently, (ii) construction of a convex optimization problem through a non-linear transformation, (iii) building a piecewise linear approximation of the non-linear terms in the problem. Additional contributions of this paper include proofs of approximation bounds, detailed experimental results showing the advantages of GOSAQ and PASAQ in solution quality over the benchmark algorithm (BRQR) and the efficiency of PASAQ. Given these results, PASAQ is at the heart of the PROTECT system, which is deployed for the US Coast Guard in the port of Boston, and is now headed to other ports.

Categories and Subject Descriptors

H.4 [**Computing Methodology**]: Game Theory

General Terms

Algorithm, Security

Keywords

Game Theory, Human Behavior, Optimization, Quantal Response

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

1. INTRODUCTION

The recent real-world applications of attacker-defender Stackelberg security games, ARMOR, IRIS [7] and GUARDS [12], provide software assistants that help security agencies optimize allocations of their limited security resources. These applications require efficient algorithms that derive mixed (randomized) strategies for the defender (security agencies), taking into account an attacker's surveillance and best response. The algorithms underlying these applications [7] or most others in the literature [1, 10] have assumed perfect rationality of the human attacker, who strictly maximizes his expected utility. While this is a standard game-theoretic assumption and appropriate as an approximation in first generation applications, it is a well-accepted limitation of classical game theory [4]. Indeed, algorithmic solutions based on this assumption may not be robust to the boundedly rational decision making of a human adversary (leading to reduced expected defender reward), and may also be limited in exploiting human biases.

To address this limitation, several models have been proposed to capture human bounded rationality in game-theoretic settings [14, 5, 11]. Among these, the quantal response (QR) model [11] is an important solution concept. QR assumes errors in human decision making and suggests that instead of strictly maximizing utility, individuals respond stochastically in games: the chance of selecting a non-optimal strategy increases as the associated cost decreases. The QR model has received widespread support in the literature in terms of its superior ability to model human behavior in games [6, 14], including in recent multi-agent systems literature [17]. An even more relevant study in the context of security games showed that defender security allocations assuming a quantal response model of adversary behavior outperformed several competing models in experiments with human subjects [18]. QR is among the best-performing current models (with significant support in the literature) and one that allows tuning of the 'adversary rationality level' as explained later. Hence this model is one that can be practically used by security agencies desiring to not be locked into adversary models of perfect rationality.

Unfortunately, in computing optimal defender strategies in security games assuming an adversary with quantal response (QR-adversary), we face two major difficulties: (1) solving a nonlinear non-convex optimization problem efficiently for massive real-world security games; and (2) addressing resource assignment constraints in security games, which adds to the complexity of computing the optimal defender strategy. Yet, scaling-up to massive security problems and handling constraints on resource assignments are essential to address real-world problems such as computing strategies for Federal Air Marshals Service (FAMS) [7] and the US Coast Guard (USCG) [13].

Yang et al. [18] introduced the algorithm BRQR to solve a Stackelberg security game with a QR-adversary. BRQR however was not guaranteed to converge to the optimal solution, as it used a nonlinear solver with multi-starts to obtain an efficient solution to a non-convex optimization problem. Furthermore, that work did not consider resource assignment constraints that are included in this paper. Nevertheless we compare the performance of the proposed algorithms against BRQR, since it is the benchmark algorithm. Another existing algorithm that efficiently computes the Quantal Response Equilibrium [15] only applies to cases where all the players have the same level of errors in their quantal response, a condition not satisfied in security games. In particular, in security games, the defender's strategy is based on a computer-aided decision-making tool, and therefore it is a best response. Adversaries, on the other hand, are human beings who may have biases and preferences in their decision making, so they are modeled with a quantal response. Therefore, new algorithms need to be developed to compute the optimal defender strategy when facing a QR-adversary in real-world security problems.

In this paper, we provide the following five contributions. First, we provide an algorithm called GOSAQ to compute the defender optimal strategy against a QR-adversary. GOSAQ uses a binary search method to iteratively estimate the global optimal solution rather than searching for it directly, which would require solving a nonlinear and non-convex fractional problem. It also uses a nonlinear variable transformation to convert the problem into a convex problem. GOSAQ leads to a ε-optimal solution, where ε can be arbitrarily small. Second, we provide another algorithm called PASAQ to approximate the optimal defender strategy. PASAQ is also based on binary search. It then converts the problem into a Mixed-Integer Linear Programming problem by using a piecewise linear approximation. PASAQ leads to an efficient approximation of the global optimal defender strategy and provides an arbitrarily near-optimal solution with a sufficiently accurate linear approximation. Third, we show that both GOSAQ and PASAQ can not only solve problems without resource assignment constraints, such as for the LAX police[7], but also problems with resource assignment constraints, such as problems for FAMS [7] and USCG [13]. Fourth, we provide the correctness/approximation-bound proof of GOSAQ and PASAQ. Fifth, we provide detailed experimental analysis on the solution quality and computational efficiency of GOSAQ and PASAQ, illustrating that both GOSAQ and PASAQ achieve better solution quality and runtime scalability than the previous benchmark algorithm BRQR [18]. Indeed, PASAQ can potentially be applied to most of the real-world deployments of the Stackelberg Security Game, including ARMOR and IRIS [7] that are based on a perfect rationality model of the adversary. This should improve the performances of such systems when dealing with human adversaries. In fact, PASAQ is at the heart of the PROTECT system [13] deployed by the US Coast Guard at the port of Boston and that is now headed to other ports in the US.

2. PROBLEM STATEMENT

We consider a Stackelberg Security Game [7, 18, 9] (SSG) with a single leader and at least one follower, where the defender plays the role of the leader and the adversary plays the role of the follower. The defender and attacker may represent organizations and need not be single individuals. We use the following notation to describe a SSG, also listed in Table 1: the defender has a total of M resources to protect a set of targets $\mathcal{T} = \{1, \ldots, |\mathcal{T}|\}$. The outcomes of the SSG depend only on whether or not the attack is successful. So given a target i, the defender receives reward R_i^d if the adversary attacks a target that is covered by the defender; otherwise, the defender receives penalty P_i^d. Correspondingly, the attacker receives penalty P_i^a in the former case; and reward R_i^a in the latter case. Note that a key property of SSG is that while the games may be non-zero-sum, $R_i^d > P_i^d$ and $R_i^a > P_i^a$, $\forall i$ [9]. In other words, adding resources to cover a target helps the defender and hurts the attacker.

Table 1: Notations used in this paper

$\mathcal{T}$	Set of targets; $i \in \mathcal{T}$ denotes target i
x_i	Probability that target i is covered by a resource
R_i^d	Defender reward for covering i if it's attacked
P_i^d	Defender penalty on not covering i if it's attack
R_i^a	Attacker reward for attacking i if it's not covered
P_i^a	Attacker penalty on attacking i if it's covered
$\mathcal{A}$	Set of defender strategies; $A_j \in \mathcal{A}$ denotes j^{th} strategy
a_j	Probability for defender to choose strategy A_j
M	Total number of resources

We denote the j^{th} individual defender strategy as A_j, which is an assignment of all the security resources. Generally, we could represent A_j as a column vector $A_j = \langle A_{ij} \rangle^T$, where A_{ij} indicates whether or not target i is covered by assignment j. Let $\mathcal{A} = \{A_j\}$ be the set of feasible assignments of resources and let a_j be the probability of selecting strategy j. Given this probability of selecting defender strategies we can compute the likelihood of protecting any specific target i as the *marginal* $x_i = \sum_{A_j \in \mathcal{A}} a_j A_{ij}$. The marginals x_i clearly sum to M, the total number of resources [8, 18]. Previous work [7] has shown that defender strategies in SSGs can be represented in terms of these marginals, leading to more concise equivalent representations. In particular, the defender's expected utility if the adversary attacks target i can be written as:

$$U_i^d(x_i) = x_i R_i^d + (1 - x_i) P_i^d$$

and the adversary's expected utility on attacking target i is

$$U_i^a(x_i) = x_i P_i^a + (1 - x_i) R_i^a$$

These marginal coverage vectors can be converted to a mixed strategy over actual defender strategies when there are no resource constraints [8], such as in ARMOR [7].

In the presence of constraints on assignments of resources, we may end up with marginals that cannot be converted to probabilities over individual strategies [8]. However, as Section 2.2 shows, we can address this difficulty if we have a complete description of defender strategies set $\mathcal{A}$. In this case we can add constraints enforcing that the marginals are obtained from a convex combination of these feasible defender strategies.

In SSGs, our goal is to compute a mixed strategy for the leader to commit to based on her knowledge of the adversary's response. More specifically, given that the defender has limited resources (e.g., she may need to protect 8 targets with 3 guards), she must design her strategy to optimize against the adversary's response to maximize effectiveness.

2.1 Optimal Strategy against Quantal Response

In this work, we assume a QR-adversary, i.e. with a quantal response $\langle q_i, i \in \mathcal{T} \rangle$ [11] to the defender's mixed strategy $\boldsymbol{x} = \langle x_i, i \in \mathcal{T} \rangle$. The value q_i is the probability that adversary attacks target i, computed as

$$q_i(\boldsymbol{x}) = \frac{e^{\lambda U_i^a(x_i)}}{\sum_{k \in \mathcal{T}} e^{\lambda U_k^a(x_k)}} \tag{1}$$

where $\lambda \geq 0$ is the parameter of the quantal response model [11], which represents the error level in adversary's quantal response. Simultaneously, the defender maximizes her utility (given her computer-aided decision making tool):

$$U^d(\boldsymbol{x}) = \sum_{i\in\mathcal{T}} q_i(\boldsymbol{x}) U_i^d(x_i)$$

Therefore, in domains without constraints on assigning the resources, the problem of computing the optimal defender strategy against a QR-adversary can be written in terms of marginals as:

$$\texttt{P1:}\begin{cases} \max\limits_{\boldsymbol{x}} \dfrac{\sum_{i\in\mathcal{T}} e^{\lambda R_i^a} e^{-\lambda(R_i^a-P_i^a)x_i}((R_i^d-P_i^d)x_i+P_i^d)}{\sum_{i\in\mathcal{T}} e^{\lambda R_i^a} e^{-\lambda(R_i^a-P_i^a)x_i}} \\ \text{s.t.} \sum\limits_{i\in\mathcal{T}} x_i \le M \\ 0 \le x_i \le 1, \quad \forall i \in \mathcal{T} \end{cases}$$

Problem P1 has a polyhedral feasible region and is a non-convex fractional objective function.

2.2 Resource Assignment Constraint

In many real world security problems, there are constraints on assigning the resources. For example, in the FAMS problem [7], an air marshal is scheduled to protect 2 flights (targets) out of M total flights. The total number of possible schedule is $\binom{M}{2}$. However, not all of the schedules are feasible, since the flights scheduled for an air marshal have to be connected, e.g. an air marshal cannot be on a flight from A to B and then on a flight C to D. A resource assignment constraint implies that the feasible assignment set $\mathcal{A}$ is restricted; not all combinatorial assignment of resources to targets are allowed. Hence, the marginals on targets, $\boldsymbol{x}$, are also restricted.

Definition 1. *We consider a marginal coverage* $\mathbf{x}$ *to be feasible if and only if there exists* $a_j \ge 0, A_j \in \mathcal{A}$ *such that* $\sum_{A_j\in\mathcal{A}} a_j = 1$ *and for all* $i \in \mathcal{T}$, $x_i = \sum_{A_j\in\mathcal{A}} a_j A_{ij}$.

In fact, $\langle a_j \rangle$ is the mixed strategy over all the feasible assignments of the resources. In order to compute the defender's optimal strategies against a QR-adversary in the presence of resource-assignment constraints, we need to solve P2. The constraints in P1 are modified to enforce feasibility of the marginal coverage.

$$\texttt{P2:}\begin{cases} \max\limits_{\boldsymbol{x},a} \dfrac{\sum_{i\in\mathcal{T}} e^{\lambda R_i^a} e^{-\lambda(R_i^a-P_i^a)x_i}((R_i^d-P_i^d)x_i+P_i^d)}{\sum_{i\in\mathcal{T}} e^{\lambda R_i^a} e^{-\lambda(R_i^a-P_i^a)x_i}} \\ \text{s.t.} \sum\limits_{i\in\mathcal{T}} x_i \le M \\ x_i = \sum\limits_{A_j\in\mathcal{A}} a_j A_{ij}, \quad \forall i \in \mathcal{T} \\ \sum\limits_{A_j\in\mathcal{A}} a_j = 1 \\ 0 \le a_j \le 1, \quad \forall A_j \in \mathcal{A} \end{cases}$$

3. BINARY SEARCH METHOD

We need to solve P1 and P2 to compute the optimal defender strategy, which requires optimally solving a non-convex problem which is in general an NP-hard problem [16]. In this section, we describe the basic structure of using a binary search method to solve the two problems. However, further efforts are required to convert this skeleton into actual efficiently runnable algorithms. We will fill in the additional details in the next two sections.

For notational simplicity, we first define the symbols $\forall\ i \in \mathcal{T}$ in Table 2. We then denote the numerator and denominator of the objective function in P1 and P2 by $N(\boldsymbol{x})$ and $D(\boldsymbol{x})$:

Table 2: Symbols for Targets in SSG

$\theta_i := e^{\lambda R_i^a} > 0$	$\beta_i := \lambda(R_i^a - P_i^a) > 0$	$\alpha_i := R_i^d - P_i^d > 0$

- $N(\boldsymbol{x}) = \sum_{i\in\mathcal{T}} \theta_i \alpha_i x_i e^{-\beta_i x_i} + \sum_{i\in\mathcal{T}} \theta_i P_i^d e^{-\beta_i x_i}$
- $D(\boldsymbol{x}) = \sum_{i\in\mathcal{T}} \theta_i e^{-\beta_i x_i} > 0$

The key idea of the binary search method is to iteratively estimate the global optimal value (p^*) of the fractional objective function of P1, instead of searching for it directly. Let $\mathscr{X}_f$ be the feasible region of P1 (or P2). Given a real value r, we can know whether or not $r \le p^*$ by checking

$$\exists \boldsymbol{x} \in \mathscr{X}_f, \text{ s.t. } rD(\boldsymbol{x}) - N(\boldsymbol{x}) \le 0 \tag{2}$$

We now justify the correctness of the binary search method to solve any generic fractional programming problem $\max_{\boldsymbol{x}\in\mathcal{X}_f} N(\boldsymbol{x})/D(\boldsymbol{x})$ for any functions $N(\boldsymbol{x})$ and $D(\boldsymbol{x}) > 0$.

Lemma 1. *For any real value* $r \in \mathscr{R}$, *one of the following two conditions holds.*

(a) $r \le p^* \Longleftrightarrow \exists \mathbf{x} \in \mathscr{X}_f$, *s.t.,* $rD(\mathbf{x}) - N(\mathbf{x}) \le 0$

(b) $r > p^* \Longleftrightarrow \forall \mathbf{x} \in \mathscr{X}_f$, $rD(\mathbf{x}) - N(\mathbf{x}) > 0$

PROOF. We only prove (a) as (b) is proven similarly. **'⇐':** since $\exists x$ such that $rD(\boldsymbol{x}) \le N(\boldsymbol{x})$, this means that $r \le \frac{N(\boldsymbol{x})}{D(\boldsymbol{x})} \le p^*$;

'⇒': Since P1 optimizes a continuous objective over a closed convex set, then there exists an optimal solution $\boldsymbol{x}^*$ such that $p^* = \frac{N(\boldsymbol{x}^*)}{D(\boldsymbol{x}^*)} \ge r$ which rearranging gives the result. □

Algorithm 1 describes the basic structure of the binary search method. Given the payoff matrix (P_M) and the total number of security resources ($numRes$), Algorithm 1 first initializes the upper bound (U_0) and lower bound (L_0) of the defender expected utility on Line 2. Then, in each iteration, r is set to be the mean of U and L. Line 6 checks whether the current r satisfies Equation (2). If so, $p^* \ge r$, the lower-bound of the binary search needs to be increased; in this case, it also returns a valid strategy x^r. Otherwise, $p^* < r$, the upper-bound of the binary search should be decreased. The search continues until the upper-bound and lower-bound are sufficiently close, i.e. $U - L < \epsilon$. The number of iterations in Algorithm 1 is bounded by $O(\log(\frac{U_0-L_0}{\epsilon}))$. Specifically for SSGs we can estimate the upper and lower bounds as follows:

Algorithm 1: Binary Search

```
1  Input: $\epsilon$, $P_M$ and $numRes$;
2  $(U_0, L_0) \leftarrow$ EstimateBounds($P_M, numRes$);
3  $(U, L) \leftarrow (U_0, L_0)$;
4  while $U - L \ge \epsilon$ do
5      $r \leftarrow \frac{U+L}{2}$;
6      $(feasible, \boldsymbol{x}^r) \leftarrow$ CheckFeasibility($r$);
7      if $feasible$ then
8          $L \leftarrow r$
9      else
10         $U \leftarrow r$
11 return $L, \boldsymbol{x}^L$;
```

Lower bound: Let s_u be any feasible defender strategy. The defender utility based on using s_u against a adversary's quantal response is a lower bound of the optimal solution of P1. A simple example of s_u is the uniform strategy.

Upper bound: Since $P_i^d \leq U_i^d \leq R_i^d$ we have $U_i^d \leq \max_{i \in \mathcal{T}} R_i^d$. The defender's utility is computed as $\sum_{i \in \mathcal{T}} q_i U_i^d$, where U_i^d is the defender utility on target i and q_i is the probability that the adversary attacks target i. Thus, the maximum R_i^d serves as an upper bound of U_i^d.

We now turn to feasibility checking, which is performed in Step 6 in Algorithm 1. Given a real number $r \in \mathscr{R}$, in order to check whether Equation (2) is satisfied, we introduce CF-OPT.

$$\texttt{CF-OPT:} \quad \min_{\boldsymbol{x} \in \mathscr{X}_f} rD(\boldsymbol{x}) - N(\boldsymbol{x})$$

Let δ^* be the optimal objective function of the above optimization problem. If $\delta^* \leq 0$, Equation (2) must be true. Therefore, by solving the new optimization problem and checking if $\delta^* \leq 0$, we can answer if a given r is larger or smaller than the global maximum. However, the objective function in CF-OPT is still non-convex, therefore, solving it directly is still a hard problem. We introduce two methods to address this in the next two sections.

4. GOSAQ: ALGORITHM 1 + VARIABLE SUBSTITUTION

We now present Global Optimal Strategy Against Quantal response (GOSAQ), which adapts Algorithm 1 to efficiently solve problems P1 and P2. It does so through the following nonlinear invertible change of variables:

$$y_i = e^{-\beta_i x_i}, \forall i \in \mathcal{T} \tag{3}$$

4.1 GOSAQ with No Assignment Constraint

We first focus on applying GOSAQ to solve P1 for problems with no resource assignment constraints. Here, GOSAQ uses Algorithm 1, but with a *rewritten* CF-OPT as follows given the above variable substitution:

$$\begin{aligned}
\min_{\boldsymbol{y}} \quad & r\sum_{i \in \mathcal{T}} \theta_i y_i - \sum_{i \in \mathcal{T}} \theta_i P_i^d y_i + \sum_{i \in \mathcal{T}} \frac{\alpha_i \theta_i}{\beta_i} y_i \ln(y_i) \\
\text{s.t.} \quad & \sum_{i \in \mathcal{T}} \frac{-1}{\beta_i} \ln(y_i) \leq M \qquad (4) \\
& e^{-\beta_i} \leq y_i \leq 1, \qquad \forall i \qquad (5)
\end{aligned}$$

Let's refer to the above optimization problem as GOSAQ-CP.

Lemma 2. *Let $Obj_{CF}(\mathbf{x})$ and $Obj_{GC}(\mathbf{y})$ be the objective function of CF-OPT and GOSAQ-CP respectively; $\mathscr{X}_f$ and $\mathscr{Y}_f$ denote the feasible domain of CF-OPT and GOSAQ-CP respectively:*

$$\min_{\mathbf{x} \in \mathscr{X}_f} Obj_{CF}(\mathbf{x}) = \min_{y \in \mathscr{Y}_f} Obj_{GC}(\mathbf{y}) \tag{6}$$

The proof, omitted for brevity, follows from the variable substitution in equation 6. Lemma 2 indicates that solving GOSAQ-CP is equivalent to solving CF-OPT. We now show that GOSAQ-CP is actually a convex optimization problem.

Lemma 3. *GOSAQ-CP is a convex optimization problem with a unique optimal solution.*

PROOF. We can show that both the objective function and the nonlinear constraint function (4) in GOSAQ-CP are strictly convex by taking second derivatives and showing that the Hessian matrices are positive definite. The fact that the objective is strictly convex implies that it can have only one optimal solution. □

In theory, convex optimization problems like the one above, can be solved in polynomial time through the ellipsoid method or interior point method with the volumetric barrier function [2] (in practice there are a number of nonlinear solvers capable of finding the only KKT point efficiently). Hence, GOSAQ entails running Algorithm 1, performing Step 6 with $O(\log(\frac{U_0 - L_0}{\epsilon}))$ times, and each time solving GOSAQ-CP which is polynomial solvable. Therefore, GOSAQ is a polynomial time algorithm.

We now show the bound of GOSAQ's solution quality.

Lemma 4. *Let L^* and U^* be the lower and upper bounds of GOSAQ when the algorithm stops, and $\mathbf{x}^*$ is the defender strategy returned by GOSAQ. Then,*

$$L^* \leq Obj_{P1}(\mathbf{x}^*) \leq U^*$$

where $Obj_{P1}(\mathbf{x})$ denotes the objective function of P1.

PROOF. Given r, Let $\delta^*(r)$ be the minimum value of the objective function in GOSAQ-CP. When GOSAQ stops, we have $\delta^*(L^*) \leq 0$, because from Lines 6-8 of Algorithm 1, updating the lower bound requires it. Hence, from Lemma 2, $L^* D(\boldsymbol{x}^*) - N(\boldsymbol{x}^*) \leq 0 \Rightarrow L^* \leq \frac{N(\boldsymbol{x}^*)}{D(\boldsymbol{x}^*)}$. Similarly, $\delta^*(U^*) \geq 0 \Rightarrow U^* > \frac{N(\boldsymbol{x}^*)}{D(\boldsymbol{x}^*)}$ □

Theorem 1. *Let $\mathbf{x}^*$ be the defender strategy computed by GOSAQ,*

$$0 \leq p^* - Obj_{P1}(\mathbf{x}^*) \leq \epsilon \tag{7}$$

PROOF. p^* is the global maximum of P1, so $p^* \geq Obj_{P1}(\boldsymbol{x}^*)$. Let L^* and U^* be the lower and upper bound when GOSAQ stops. Based on Lemma 4, $L^* \leq Obj_{P1}(\boldsymbol{x}^*) \leq U^*$. Simultaneously, Algorithm 1 indicates that $L^* \leq p^* \leq U^*$.

Therefore, $0 \leq p^* - Obj_{P1}(\boldsymbol{x}^*) \leq U^* - L^* \leq \epsilon$ □

Theorem 1 indicates that the solution obtained by GOSAQ is an ϵ-optimal solution.

4.2 GOSAQ with Assignment Constraints

In order to address the assignment constraints, we need to solve P2. Note that the objective function of P2 is the same as that of P1. The difference lies in the extra constraints which enforce the marginal coverage to be feasible. Therefore we once again use Algorithm 1 with variable substitution given in Equation 3, but modify GOSAQ-CP as follows (which is referred as GOSAQ-CP-C) to incorporate the extra constraints:

$$\begin{aligned}
\min_{\boldsymbol{y}, a} \quad & r\sum_{i \in \mathcal{T}} \theta_i y_i - \sum_{i \in \mathcal{T}} \theta_i P_i^d y_i + \sum_{i \in \mathcal{T}} \frac{\alpha_i \theta_i}{\beta_i} y_i \ln(y_i) \\
\text{s.t.} \quad & \texttt{Constraint } (4), (5) \\
& \frac{-1}{\beta_i} \ln(y_i) = \sum_{A_j \in \mathcal{A}} a_j A_{ij}, \quad \forall i \in \mathcal{T} \qquad (8) \\
& \sum_{A_j \in \mathcal{A}} a_j = 1 \qquad (9) \\
& 0 \leq a_j \leq 1, \quad A_j \in \mathcal{A} \qquad (10)
\end{aligned}$$

Equation (8) is a nonlinear equality constraint that makes this optimization problem non-convex. There are no known polynomial time algorithms for generic non-convex optimization problems, which can have multiple local minima. We can attempt to solve such non-convex problems using one of the efficient nonlinear solvers but we would obtain a KKT point which can be only locally optimal. There are a few research grade global solvers for non-convex programs, however they are limited to solving specific problems or small instances. Therefore, in the presence of assignment constraints, GOSAQ is no longer guaranteed to return the optimal solution as we might be left with locally optimal solutions when solving the subproblems GOSAQ-CP-C.

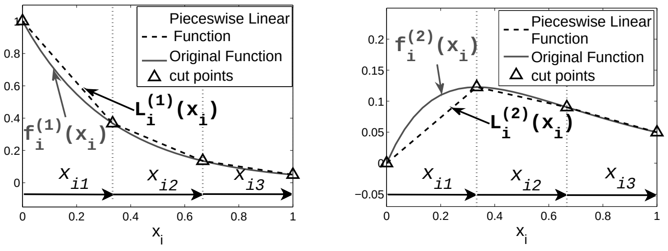
5. PASAQ: ALGORITHM 1 + LINEAR APPROXIMATION

Since GOSAQ may be unable to provide a quality bound in the presence of assignment constraints (and as shown later, may turn out to be inefficient in such cases), we propose the Piecewise linear Approximation of optimal Strategy Against Quantal response (PASAQ). PASAQ is an algorithm to compute the approximate optimal defender strategy. PASAQ has the same structure as Algorithm 1. The key idea in PASAQ is to use a piecewise linear function to approximate the nonlinear objective function in `CF-OPT`, and thus convert it into a Mixed-Integer Linear Programming (MILP) problem. Such a problem can easily include assignment constraints giving an approximate solution for a SSG against a QR-adversary with assignment constraints.

In order to demonstrate the piecewise approximation in PASAQ, we first rewrite the nonlinear objective function of `CF-OPT` as:

$$\sum_{i\in\mathcal{T}} \theta_i (r - P_i^d) e^{-\beta_i x_i} + \sum_{i\in\mathcal{T}} \theta_i \alpha_i x_i e^{-\beta_i x_i}$$

The goal is to approximate the two nonlinear function $f_i^{(1)}(x_i) = e^{-\beta_i x_i}$ and $f_i^{(2)}(x_i) = x_i e^{-\beta_i x_i}$ as two piecewise linear functions in the range $x_i \in [0,1]$, for each $i = 1..|\mathcal{T}|$. We first uniformly



(a) approximation of $f_i^{(1)}(x_i)$ (b) approximation of $f_i^{(2)}(x_i)$

Figure 1: Piecewise Linear Approximation

divide the range $[0,1]$ into K pieces (segments). Simultaneously, we introduce a set of new variables $\{x_{ik}, k = 1..K\}$ to represent the portion of x_i in each of the K pieces, $\{[\frac{k-1}{K}, \frac{k}{K}], k = 1..K\}$. Therefore, $x_{ik} \in [0, \frac{1}{K}], \forall k = 1..K$ and $x_i = \sum_{k=1}^{K} x_{ik}$. In order to ensure that $\{x_{ik}\}$ is a valid partition of x_i, all x_{ik} must satisfy: $x_{ik} > 0$ only if $x_{ik'} = \frac{1}{K}, \forall k' < k$. In other words, x_{ik} can be non-zero only when all the previous pieces are completely filled. Figures 1(a) and 1(b) display two examples of such a partition.

Thus, we can represent the two nonlinear functions as piecewise linear functions using $\{x_{ik}\}$. Let $\{(\frac{k}{K}, f_i^{(1)}(\frac{k}{K})), k = 0..K\}$ be the $K+1$ cut-points of the linear segments of function $f_i^{(1)}(x_i)$, and $\{\gamma_{ik}, k = 1..K\}$ be the slopes of each of the linear segments. Starting from $f_i^{(1)}(0)$, the piecewise linear approximation of $f_i^{(1)}(x_i)$, denoted as $L_i^{(1)}(x_i)$:

$$L_i^{(1)}(x_i) = f_i^{(1)}(0) + \sum_{k=1}^{K} \gamma_{ik} x_{ik} = 1 + \sum_{k=1}^{K} \gamma_{ik} x_{ik}$$

Similarly, we can obtain the piecewise linear approximation of $f_i^{(2)}(x_i)$, denoted as $L_i^{(2)}(x_i)$:

$$L_i^{(2)}(x_i) = f_i^{(2)}(0) + \sum_{k=1}^{K} \mu_{ik} x_{ik} = \sum_{k=1}^{K} \mu_{ik} x_{ik}$$

where, $\{\mu_{ik}, k = 1..K\}$ is the slope of each linear segment.

Table 3: Notations for Error Bound Proof

$\underline{\theta} := \min_{i\in\mathcal{T}} \theta_i$	$\overline{R^d} := \max_{i\in\mathcal{T}} \lvert R_i^d \rvert$	$\overline{\beta} := \max_{i\in\mathcal{T}} \beta_i$
$\overline{\theta} := \max_{i\in\mathcal{T}} \theta_i$	$\overline{P^d} := \max_{i\in\mathcal{T}} \lvert P_i^d \rvert$	$\overline{\alpha} := \max_{i\in\mathcal{T}} \alpha_i$

5.1 PASAQ with No Assignment Constraint

In domains without assignment constraints, PASAQ consists of Algorithm 1, but with `CF-OPT` rewritten as follows:

$$\min_{x,z} \sum_{i\in\mathcal{T}} \theta_i (r - P_i^d)(1 + \sum_{k=1}^{K} \gamma_{ik} x_{ik}) + \sum_{i\in\mathcal{T}} \theta_i \alpha_i \sum_{k=1}^{K} \mu_{ik} x_{ik}$$

$$\text{s.t.} \sum_{i\in\mathcal{T}} \sum_{k=1}^{K} x_{ik} \le M \tag{11}$$

$$0 \le x_{ik} \le \frac{1}{K}, \quad \forall i, \quad k = 1 \dots K \tag{12}$$

$$z_{ik} \frac{1}{K} \le x_{ik}, \quad \forall i, \quad k = 1 \dots K-1 \tag{13}$$

$$x_{i(k+1)} \le z_{ik}, \quad \forall i, \quad k = 1 \dots K-1 \tag{14}$$

$$z_{ik} \in \{0,1\}, \quad \forall i, \quad k = 1 \dots K-1 \tag{15}$$

Let's refer to the above MILP formulation as `PASAQ-MILP`.

Lemma 5. *The feasible region for* $\mathbf{x} = \langle x_i = \sum_{k=1}^{K} x_{ik}, i \in \mathcal{T} \rangle$ *of* `PASAQ-MILP` *is equivalent to that of* `P1`

JUSTIFICATION. The auxiliary integer variable z_{ik} indicates whether or not $x_{ik} = \frac{1}{K}$. Equation (13) enforces that $z_{ik} = 0$ only when $x_{ik} < \frac{1}{K}$. Simultaneously, Equation (14) enforces that $x_{i(k+1)}$ is positive only if $z_{ik} = 1$. Hence,$\{x_{ik}, k = 1..K\}$ is a valid partition of x_i and $x_i = \sum_{k=1}^{K} x_{ik}$ and that $x_i \in [0,1]$. Thus, the feasible region of `PASAQ-MILP` is equivalent to `P1`

Lemma 5 shows that the solution provided by PASAQ is in the feasible region of `P1`. However, PASAQ approximates the minimum value of `CF-OPT` by using `PASAQ-MILP`, and furthermore solves `P1` approximately using binary search. Hence, we need to show an error bound on the solution quality of PASAQ.

We first show Lemma 6, 7 and 8 on the way to build the proof for the error bound. Due to space constraints, many proofs are abbreviated; full proofs are available in an on-line appendix[1]. Further, we define two constants which are decided by the game payoffs: $C_1 = (\overline{\theta}/\underline{\theta}) e^{\overline{\beta}} \{(\overline{R^d} + \overline{P^d})\overline{\beta} + \overline{\alpha}\}$ and $C_2 = 1 + (\overline{\theta}/\underline{\theta}) e^{\overline{\beta}}$. The notation used is defined in Table 3. In the following, we are interested in obtaining a bound on the difference between p^* (the global optimal obtained from `P1`) and $Obj_{P1}(\tilde{\mathbf{x}}^*)$, where $\tilde{\mathbf{x}}^*$ is the strategy obtained from PASAQ. However, along the way, we have to obtain a bound for the difference between $Obj_{P1}(\tilde{\mathbf{x}}^*)$ and its corresponding piecewise linear approximation $\tilde{Obj}_{P1}(\tilde{\mathbf{x}}^*)$.

Lemma 6. *Let* $\tilde{N}(\mathbf{x}) = \sum_{i\in\mathcal{T}} \theta_i \alpha_i L_i^{(2)}(x_i) + \sum_{i\in\mathcal{T}} \theta_i P_i^d L_i^{(1)}(x_i)$ *and* $\tilde{D}(\mathbf{x}) = \sum_{i\in\mathcal{T}} \theta_i L_i^{(1)}(x_i) > 0$ *be the piecewise linear approximation of* $N(\mathbf{x})$ *and* $D(\mathbf{x})$ *respectively. Then,* $\forall \mathbf{x} \in \mathscr{X}_f$

$$|N(\mathbf{x}) - \tilde{N}(\mathbf{x})| \le (\overline{\theta}\overline{\alpha} + \overline{P^d}\overline{\theta}\overline{\beta}) \frac{|\mathcal{T}|}{K}$$

$$|D(\mathbf{x}) - \tilde{D}(\mathbf{x})| \le \overline{\theta}\overline{\beta} \frac{|\mathcal{T}|}{K}$$

[1]http://anon-aamas2012.webs.com/FullProof.pdf

Lemma 7. *The difference between the objective funciton of* P1*, $Obj_{P1}(\mathbf{x})$, and its corresponding piecewise linear approximation, $\tilde{Obj}_{P1}(\mathbf{x})$, is less than $C_1\frac{1}{K}$*

PROOF.

$$|Obj_{P1}(\boldsymbol{x}) - \tilde{Obj}_{P1}(\boldsymbol{x})| = |\frac{N(\boldsymbol{x})}{D(\boldsymbol{x})} - \frac{\tilde{N}(\boldsymbol{x})}{\tilde{D}(\boldsymbol{x})}|$$

$$= |\frac{N(\boldsymbol{x})}{D(\boldsymbol{x})} - \frac{N(\boldsymbol{x})}{\tilde{D}(\boldsymbol{x})} + \frac{N(\boldsymbol{x})}{\tilde{D}(\boldsymbol{x})} - \frac{\tilde{N}(\boldsymbol{x})}{\tilde{D}(\boldsymbol{x})}|$$

$$\leq \frac{1}{\tilde{D}(\boldsymbol{x})}(|Obj_{P1}(\boldsymbol{x})||D(\boldsymbol{x}) - \tilde{D}(\boldsymbol{x})| + |N(\boldsymbol{x}) - \tilde{N}(\boldsymbol{x})|)$$

Based on Lemma 6, $|Obj_{P1}(\boldsymbol{x})| \leq \overline{R^d}$, and $\tilde{D}(\boldsymbol{x}) \geq |\mathcal{T}|\underline{\theta}e^{-\overline{\beta}}$.

$$|Obj_{P1}(\boldsymbol{x}) - \tilde{Obj}_{P1}(\boldsymbol{x})| \leq C_1\frac{1}{K} \qquad \square$$

Lemma 8. *Let $\tilde{L}^*$ and L^* be final lower bound of* PASAQ *and* GOSAQ*,*

$$L^* - \tilde{L}^* \leq C_1\frac{1}{K} + C_2\epsilon$$

Lemma 9. *Let $\tilde{L}^*$ and $\tilde{U}^*$ be the final lower and upper bounds of* PASAQ*, and $\tilde{\mathbf{x}}^*$ is the defender strategy returned by* PASAQ*. Then,*

$$\tilde{L}^* \leq \tilde{Obj}_{P1}(\tilde{\mathbf{x}}^*) \leq \tilde{U}^*$$

Theorem 2. *Let $\tilde{x}^*$ be the defender strategy computed by* PASAQ*, p^* is the global optimal defender expected utility,*

$$0 \leq p^* - Obj_{P1}(\tilde{x}^*) \leq 2C_1\frac{1}{K} + (C_2+1)\epsilon$$

PROOF. The first inequality is implied since $\tilde{x}^*$ is a feasible solution. Furthermore,

$$p^* - Obj_{P1}(\tilde{x}^*) = (p^* - L^*) + (L^* - \tilde{L}^*) + (\tilde{L}^* - \tilde{Obj}_{P1}(\tilde{x}^*)) + (\tilde{Obj}_{P1}(\tilde{x}^*) - Obj_{P1}(\tilde{x}^*))$$

Algorithm 1 indicates that $L^* \leq p^* \leq U^*$, hence $p^* - L^* \leq \epsilon$. Additionally, Lemma 7, 8 and 9 provide an upper bound on $\tilde{Obj}_{P1}(\tilde{x}^*) - Obj_{P1}(\tilde{x}^*)$, $L^* - \tilde{L}^*$ and $\tilde{L}^* - \tilde{Obj}_{P1}(\tilde{x}^*)$, therefore

$$p^* - Obj_{P1}(\tilde{x}^*) \leq \epsilon + C_1\frac{1}{K} + C_2\epsilon + C_1\frac{1}{K} \leq 2C_1\frac{1}{K} + (C_2+1)\epsilon \quad \square$$

Theorem 2 suggests that, given a game instance, the solution quality of PASAQ is bounded linearly by the binary search threshold ϵ and the piecewise linear accuracy $\frac{1}{K}$. Therefore the PASAQ solution can be made arbitrarily close to the optimal solution with sufficiently small ϵ and sufficiently large K.

5.2 PASAQ With Assignment Constraints

In order to extend PASAQ to handle the assignment constraints, we need to modify `PASAQ-MILP` as the follows, referred to as `PASAQ-MILP-C`,

$$\min_{x,z,a} \sum_{i\in\mathcal{T}} \theta_i(r - P_i^d)(1 + \sum_{k=1}^{K}\gamma_{ik}x_{ik}) + \sum_{i\in\mathcal{T}}\theta_i\alpha_i\sum_{k=1}^{K}\mu_{ik}x_{ik}$$

s.t. `Constraint` $(11) - (15)$

$$\sum_{k=1}^{K} x_{ik} = \sum_{A_j\in\mathcal{A}} a_j A_{ij}, \quad \forall i \in \mathcal{T} \qquad (16)$$

$$\sum_{A_j\in\mathcal{A}} a_j = 1 \qquad (17)$$

$$0 \leq a_j \leq 1, \quad A_j \in \mathcal{A} \qquad (18)$$

`PASAQ-MILP-C` is an MILP so it can be solved optimally with any MILP solver (e.g. CPLEX). We can prove, similarly as we did for Lemma 5, that the above MILP formulation has the same feasible region as `P2`. Hence, it leads to a feasible solution of `P2`. Furthermore, the error bound of PASAQ relies on the approximation accuracy of the objective function by the piecewise linear function and the fact that the subproblem `PASAQ-MILP-C` can be solved optimally. Both conditions have not changed from the cases without assignment constraints to the cases with assignment constraints. Hence, the error bound is the same as that shown in Theorem 2.

6. EXPERIMENTS

We separate our experiments into two sets: the first set focuses on the cases where there is no constraint on assigning the resources; the second set focuses on cases with assignment constraints. In both sets, we compare the solution quality and runtime of the two new algorithms, GOSAQ and PASAQ, with the previous benchmark algorithm BRQR. The results were obtained using CPLEX to solve the MILP for PASAQ. For both BRQR and GOSAQ, we use the MATLAB toolbox function `fmincon` to solve nonlinear optimization problems[2]. All experiments were conducted on a standard 2.00GHz machine with 4GB main memory. For each setting of the experiment parameters (i.e. number of targets, amount of resources and number of assignment constraints), we tried 50 different game instances. In each game instance, payoffs R_i^d and R_i^a are chosen uniformly randomly from 1 to 10, while P_i^d and P_i^a are chosen uniformly randomly from -10 to -1; feasible assignments A_j are generated by randomly setting each element A_{ij} to 0 or 1. For the parameter λ of the quantal response in Equation (1), we used the same value ($\lambda = 0.76$) as reported in [18].

6.1 No Assignment Constraints

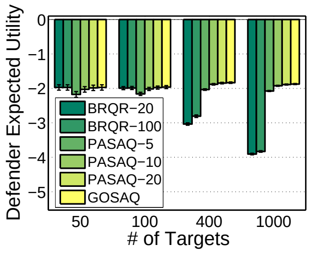
We first present experimental results comparing the solution quality and runtime of the three algorithms (GOSAQ,PASAQ and BRQR) in cases without assignment constraints.

Solution Quality: For each game instance, GOSAQ provides the ϵ-optimal defender expected utility, BRQR presents the best local optimal solution among all the local optimum it finds, and PASAQ leads to an approximated global optimal solution. We measure the solution quality of different algorithms using average defender's expected utility over all the 50 game instances.

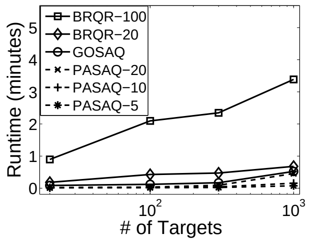
Figures 2(a), 2(c) and 2(e) show the solution quality results of different algorithms under different conditions. In all three figures, the average defender expected utility is displayed on the y-axis. On the x-axis, Figure 2(a) changes the numbers of targets ($|\mathcal{T}|$) keeping the ratio of resources (M) to targets and ϵ fixed as shown in the caption; Figure 2(c) changes the ratio of resources to targets fixing targets and ϵ as shown; and Figure 2(e) changes the value of the binary search threshold ϵ. Given a setting of the parameters ($|\mathcal{T}|$, M and ϵ), the solution qualities of different algorithms are displayed in a group of bars. For example, in Figure 2(a), $|\mathcal{T}|$ is set to 50 for the leftmost group of bars, M is 5 and $\epsilon = 0.01$. From left to right, the bars show the solution quality of BRQR (with 20 and 100 iterations), PASAQ (with 5,10 and 20 pieces) and GOSAQ.

Key observations from Figures 2(a), 2(c) and 2(e) include: (i) The solution quality of BRQR drops quickly as the number of targets increases; increasing the number of iterations in BRQR improves the solution quality, but the improvement is very small. (ii) The solution quality of PASAQ improves as the number of pieces increases;

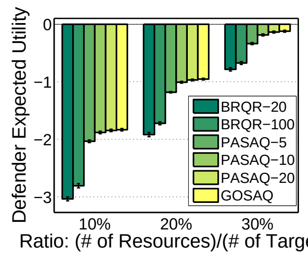
[2]We also tried the KNITRO [3] solver. While it gave the same solution quality as `fmincon`, it was three-times slower than `fmincon`; as a result we report results with `fmincon`



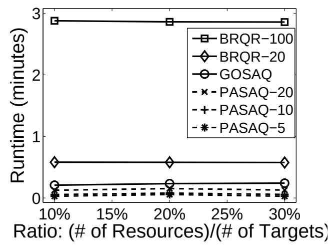
(a) Solution Quality v.s. $|\mathcal{T}|$ ($M = 0.1|\mathcal{T}|$, $\epsilon = 0.01$)



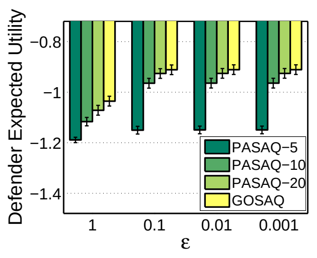
(b) Runtime v.s. $|\mathcal{T}|$ ($M = 0.1|\mathcal{T}|$, $\epsilon = 0.01$)



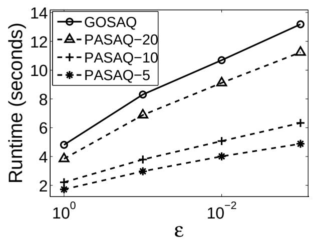
(c) Solution Quality v.s. M ($|\mathcal{T}| = 400$, $\epsilon = 0.01$)



(d) Runtime v.s. M ($|\mathcal{T}| = 400$, $\epsilon = 0.01$)



(e) Solution Quality v.s. ϵ ($|\mathcal{T}| = 400$, $M = 80$)



(f) Runtime v.s. ϵ ($|\mathcal{T}| = 400$, $M = 80$)

Figure 2: Solution Quality and Runtime Comparison, without assignment constraints (better in color)

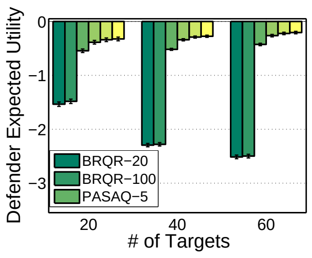
and it converges to the GOSAQ solution as the number of pieces becomes larger than 10. (iii) As the number of resources increases, the defender expected utility also increases; and the resource count does not impact the relationship of solution quality between different algorithms. (iv) As ϵ becomes smaller, the solution quality of both GOSAQ and PASAQ improves. However, after epsilon becomes sufficiently small (≤ 0.1), no substantial improvement is achieved by further decreasing the value of ϵ. In other words, the solution quality of both GOSAQ and PASAQ converges.

In general, BRQR has the worst solution quality; GOSAQ has the best solution quality. PASAQ achieves almost the same solution quality as GOSAQ when it uses more than 10 pieces.

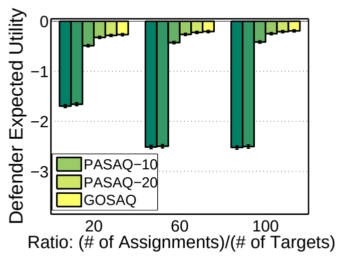
Runtime: We present the runtime results in Figures 2(b), 2(d) and 2(f). In all three figures, the y-axis display the runtime, the x-axis displays the variables which we vary to measure their impact on the runtime of the algorithms. For BRQR run time is the sum of the run-time across all its iterations.

Figure 2(b) shows the change in runtime as the number of targets increases. The number of resources and the value of ϵ are shown in the caption. BRQR with 100 iterations is seen to run significantly slower than GOSAQ and PASAQ. Figure 2(d) shows the impact of the ratio of resource to targets on the runtime. The figure indicates that the runtime of the three algorithms is independent of the change in the number of resources. Figure 2(f) shows how runtime of GOSAQ and PASAQ is affected by the value of ϵ. On the x-axis, the value for ϵ decreases from left to right. The runtime increases linearly as ϵ decreases exponentially. In both Figures 2(d) and 2(f), the number of targets and resources are displayed in the caption.

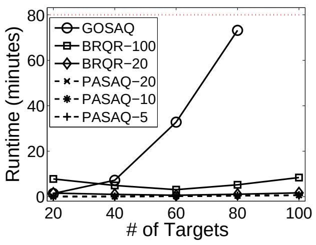
Overall, the results suggest that GOSAQ is the algorithm of choice when the domain has no assignment constraints. Clearly, BRQR has the worst solution quality, and it is the slowest of the set of algorithms. PASAQ has a solution quality that approaches that of GOSAQ when the number of pieces is sufficiently large (≥ 10), and GOSAQ and PASAQ also achieve comparable runtime efficiency. Thus, in cases with no assignment constraints, PASAQ offers no advantages over GOSAQ.



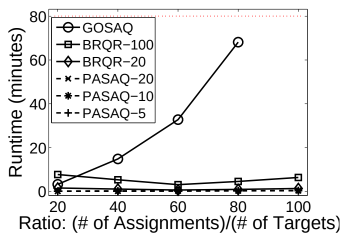
(a) Solution Quality v.s. $|\mathcal{T}|$ ($|\mathcal{A}| = 60|\mathcal{T}|$)



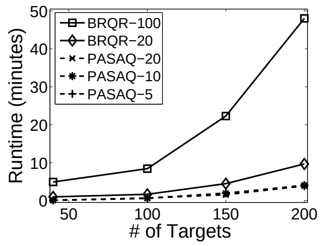
(b) Solution Quality v.s. $|\mathcal{A}|$ ($|\mathcal{T}| = 60$)



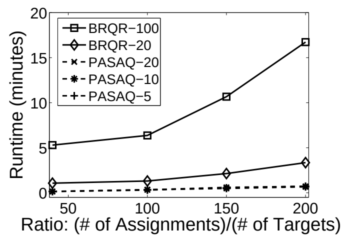
(c) Runtime v.s. $|\mathcal{T}|$ ($|\mathcal{A}| = 60|\mathcal{T}|$)



(d) Runtime v.s. $|\mathcal{A}|$ ($|\mathcal{T}| = 60$)



(e) Runtime v.s. $|\mathcal{T}|$ ($|\mathcal{A}| = 60|\mathcal{T}|$)



(f) Runtime v.s. $|\mathcal{A}|$ ($|\mathcal{T}| = 60$)

Figure 3: Solution Quality and Runtime Comparison, with assignment constraint (better in color)

6.2 With Assignment Constraint

In the second set, we introduce assignment constraints into the problem. The feasible assignments are randomly generated. We present experimental results on both solution quality and runtime.

Solution Quality: Figures 3(a) and 3(b) display the solution quality of the three algorithms with varying number of targets ($|\mathcal{T}|$) and varying number of feasible assignments ($|\mathcal{A}|$). In both figures, the average defender expected utility is displayed on the y-axis. In Figure 3(a) the number of targets is displayed on the x-axis, and the ratio of $|\mathcal{A}|$ to $|\mathcal{T}|$ is set to 60. BRQR is seen to have very poor performance. Furthermore, there is very little gain in solution quality from increasing its number of iterations. While GOSAQ provides the best solution quality, PASAQ achieves almost identical solution quality when the number of pieces is sufficiently large (> 10). Figure 3(b) shows how solution quality is impacted by the number of feasible assignments, which is displayed on the x-axis. Specifically, the x-axis shows numbers of assignment constraints $\mathcal{A}$ to be 20 times, 60 times and 100 times the number of targets. The number of targets is set to 60. Once again, BRQR has significantly lower solution quality, and it drops as the number of assignments increases; and PASAQ again achieves almost the same solution quality as GOSAQ, as the number the number of pieces is larger than 10.

Runtime: We present the runtime results in Figures 3(c), 3(e), 3(d) and 3(f). In all experiments, we set 80 minutes as the cutoff. Figure 3(c) displays the runtime on the y-axis and the number of targets on the x-axis. It is clear that GOSAQ runs significantly slower than both PASAQ and BRQR, and slows down exponentially as the number of targets increases. Figure 3(e) shows extended runtime result of BRQR and PASAQ as the number of targets increases. PASAQ runs in less than 4 minutes with 200 targets and 12000 feasible assignments. BRQR runs significantly slower with higher number of iterations.

Overall, the results suggest that PASAQ is the algorithm of choice when the domain has assignment constraints. Clearly, BRQR has significantly lower solution quality than PASAQ. PASAQ not only has a solution quality that approaches that of GOSAQ when the number of pieces is sufficiently large (≥ 10), PASAQ is significantly faster than GOSAQ (which suffers exponential slowdown with scale-up in the domain).

7. CONCLUSION

This paper marks an advance over the state-of-the-art in security games. It goes beyond the assumption of perfect rationality of human adversaries embedded in deployed applications [7] and most of the current algorithms [1, 10] for Stackelberg security games; instead, it models the human adversaries' bounded rationality using the quantal response (QR) model. This work overcomes the difficulties in developing efficient methods to solve the massive security games in real applications, including solving a nonlinear and non-convex optimization problem and handling constraints on assigning security resources in designing defender strategies. In addressing these difficulties, key contributions in this paper include: (i) a new algorithm, GOSAQ, which guarantees the global optimal solution in computing the defender strategy against an adversary's quantal response; (ii) an efficient approximation algorithm, PASAQ, which provides more efficient computation of the defender strategy with nearly-optimal solution quality; (iii) algorithms solving problems with resource assignment constraint; (iv) proof of correctness/approximation-error of the algorithms; (v) detailed experimental results which show that both GOSAQ and PASAQ achieve much better solution quality than the benchmark algorithm (BRQR), and that PASAQ achieves much better computational efficiency than both GOSAQ and BRQR. Given these results, PASAQ is at the heart of the PROTECT system which is currently being used for the US Coast Guard in the port of Boston, and is currently being deployed in the port of New York.

8. ACKNOWLEDGMENTS

This research was supported by Army Research Office under the grant # W911NF-10-1-0185. We also thank Rajiv Maheswaran for giving valuable comments. Fernando Ordonez would also like to acknowledge the support of grant ICM/FIC P10-024-F.

9. REFERENCES

[1] N. Basiloco, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. *In AAMAS*, 2009.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, 2004.

[3] R. H. Byrd, J. Nocedal, and R. A. Waltz. Knitro: An integrated package for nonlinear optimization. In *Large-Scale Nonlinear Optimization*, pages 35–59. G. di Pillo and M. Roma, eds, Springer-Verlag, 2006.

[4] C. F. Camerer. *Behavioral Game Theory: Experiments in Strategic Interaction*. Princeton University Press, Princeton, New Jersey, 2003.

[5] C. F. Camerer, T. Ho, and J. Chong. A cognitive hierarchy model of games. *QJE*, 119(3):861–898, 2004.

[6] P. A. Haile, A. Hortacsu, and G. Kosenok. On the empirical content of quantal response equilibrium. *The American Economic Review*, 98(1):180–200, March 2008.

[7] M. Jain, J. Pita, J. Tsai, C. Kiekintveld, S. Rathi, F. Ordonez, and M. Tambe. Software assistants for patrol planning at lax and federal air marshals service. *Interfaces*, 40(4):267–290, 2010.

[8] D. Korzhyk, V. Conitzer, and R. Parr. Complexity of computing optimal stackelberg strategies in security resource allocation games. *In AAAI*, 2010.

[9] D. Korzhyk, Z. Yin, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *In JAIR*, 2011.

[10] J. Letchford and Y. Vorobeychik. Computing randomized security strategies in networked domains. *AARM Workshop In AAAI*, 2011.

[11] R. D. McKelvey and T. R. Palfrey. Quantal response equilibria for normal form games. *Games and Economic Behavior*, 2:6–38, 1995.

[12] J. Pita, M. Tambe, C. Kiekintveld, S. Cullen, and E. Steigerwald. Guards - game theoretic security allocation on a national scale. *In AAMAS*, 2011.

[13] E. Shieh, B. An, R. Yang, M. Tambe, C. Baldwin, J. DiRenzo, B. Maule, and G. Meyer. Protect: A deployed game theoretic system to protect the ports of the united states. *In AAMAS*, 2012.

[14] D. O. Stahl and P. W. Wilson. Experimental evidence on players' models of other players. *JEBO*, 25(3):309–327, 1994.

[15] T. Turocy. A dynamic homotopy interpretation of the logistic quantal response equilibrium correspondence. *Games and Economic Behavior*, 51(2):243–263, 2006.

[16] S. A. Vavasis. Complexity issues in global optimization: a survey. In *Handbook of Global Optimization*, pages 27–41. In R. Horst and P.M. Pardalos, editors, Kluwer, 1995.

[17] J. R. Wright and K. Leyton-Brown. Beyond equilibrium: Predicting human behavior in normal-form games. *In AAAI*, 2010.

[18] R. Yang, C. Kiekintveld, F. Ordonez, M. Tambe, and R. John. Improving resource allocation strategy against human adversaries in security games. *In IJCAI*, 2011.

A Unified Method for Handling Discrete and Continuous Uncertainty in Bayesian Stackelberg Games

Zhengyu Yin and Milind Tambe
University of Southern California, Los Angeles, CA 90089, USA
{zhengyuy, tambe}@usc.edu

ABSTRACT

Given their existing and potential real-world security applications, Bayesian Stackelberg games have received significant research interest [3, 12, 8]. In these games, the defender acts as a leader, and the many different follower types model the uncertainty over discrete attacker types. Unfortunately since solving such games is an NP-hard problem, scale-up has remained a difficult challenge.

This paper scales up Bayesian Stackelberg games, providing a novel unified approach to handling uncertainty not only over discrete follower types but also other key continuously distributed real world uncertainty, due to the leader's execution error, the follower's observation error, and continuous payoff uncertainty. To that end, this paper provides contributions in two parts. First, we present a new algorithm for Bayesian Stackelberg games, called HUNTER, to scale up the number of types. HUNTER combines the following five key features: i) efficient pruning via a best-first search of the leader's strategy space; ii) a novel linear program for computing tight upper bounds for this search; iii) using Bender's decomposition for solving the upper bound linear program efficiently; iv) efficient inheritance of Bender's cuts from parent to child; v) an efficient heuristic branching rule. Our experiments show that HUNTER provides orders of magnitude speedups over the best existing methods to handle discrete follower types. In the second part, we show HUNTER's efficiency for Bayesian Stackelberg games can be exploited to also handle the continuous uncertainty using sample average approximation. We experimentally show that our HUNTER-based approach also outperforms latest robust solution methods under continuously distributed uncertainty.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithms, Security, Performance

Keywords

Game theory, Bayesian Stackelberg Games, Relaxation

1. INTRODUCTION

Due to their significance in real-world security, there has been a lot of recent research activity in leader-follower Stackelberg games, oriented towards producing deployed solutions: ARMOR at LAX [9], IRIS for Federal Air Marshals Service [9], and GUARDS for the TSA [14]. Bayesian extension to Stackelberg game has been used to model the uncertainty over players' preferences [12, 8] by allowing multiple discrete follower types, as well as, by use of sampling-based algorithms, continuous payoff uncertainty [10].

The key idea in this paper is to scale-up Bayesian Stackelberg games, providing a novel unified approach to handling not only discrete follower types but also continuous uncertainty. Scalability of discrete follower types is essential in domains such as road network security [6], where each follower type could represent a criminal attempting to follow a certain path. Scaling up the number of types is also necessary for the sampling-based algorithms to obtain high quality solutions under continuous uncertainty. Unfortunately, such scale-up remains difficult, as finding the equilibrium of a Bayesian Stackelberg game is NP-hard [5]. Indeed, despite the recent algorithmic advancement including Multiple-LPs [5], DOBSS [12], HBGS [8], none of these techniques can handle games with more than ≈ 50 types, even when the number of actions per player is as few as 5: inadequate both for scale-up in discrete follower types and for sampling-based approaches. This scale-up difficulty has led to an entirely new set of algorithms developed for handling continuous payoff uncertainty [10], and continuous observation and execution error [16]; these algorithms do not handle discrete follower types, however.

This paper provides contributions in two parts. In the first part, to address the challenge of discrete uncertainty, we propose a novel algorithm for solving Bayesian Stackelberg games, called HUNTER, combining the following five key ideas. First, it conducts a best-first search in the follower's best-response assignment space, which only expands a small number of nodes (within an exponentially large assignment space). Second, HUNTER computes tight upper bounds to speed up this search using a novel linear program. Third, HUNTER solves this linear program efficiently using Bender's decomposition. Fourth, we show that the Bender's cuts generated in a parent node are valid cuts for its children, providing further speedups. Finally, HUNTER deploys a heuristic branching rule to further improve efficiency. Thus, this paper's contribution is in combining an AI search technique (best-first search) with multiple techniques from Operations Research (disjunctive program and Bender's decomposition) to provide a novel efficient algorithm; the application of these techniques for solving Stackelberg games had not been explored earlier, and thus their application towards solving these games, as well as their particular synergistic combination in HUNTER are both novel. Our experiments show HUNTER dramatically improves the scalability of the number of types over other

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

existing approaches [12, 8].

In the second part of our contribution, we show that via sample average approximation, HUNTER for Bayesian Stackelberg games can be used in handling continuously distributed uncertainty such as the leader's execution error, the follower's observation noise, and both players' preference uncertainty. For comparison, we consider a class of Stackelberg games motivated by security applications, and *enhance* two existing robust solution methods, BRASS [13] and RECON [16] to handle such uncertainty. We again show that HUNTER provides significantly better performance than BRASS and RECON. Our final set of experiments in this paper also illustrates HUNTER's *unique ability* to handle both discrete and continuous uncertainty within a single problem.

2. BACKGROUND AND NOTATION

The first part of the paper is focused on solving Bayesian Stackelberg games with discrete follower types; this background section focuses on such games. A Stackelberg game is a two-person game played by a leader and a follower [15]. Following the recent literature [8], we focus on Stackelberg games where the leader commits to a mixed strategy first, and the follower observes the leader's strategy and responds with a pure strategy, maximizing his utility correspondingly. We generalize this set-up by extending the definition of the leader's strategy space and the leader and follower utilities in two ways beyond what was previously considered [12, 8] and by allowing for compact representation of constraints.

We assume the leader's mixed strategy is an N-dimensional real column vector $\mathbf{x} \in \mathbb{R}^N$, bounded by a polytope $A\mathbf{x} \preceq \mathbf{b}, \mathbf{x} \succeq \mathbf{0}$, which generalizes the traditional constraint of $\sum_i x_i = 1$ and allows for compact strategy representation with constraints as in [9] (although such constraints are not the focus of this paper). Second, given a leader's strategy $\mathbf{x}$, the follower maximizes his utility by choosing from J pure strategies. For each pure strategy $j = 1, \dots, J$ played by the follower, the leader gets a utility of $\boldsymbol{\mu}_j^{\mathrm{T}}\mathbf{x} + \mu_{j,0}$ and the follower gets a utility of $\boldsymbol{\nu}_j^{\mathrm{T}}\mathbf{x} + \nu_{j,0}$, where $\boldsymbol{\mu}_j, \boldsymbol{\nu}_j$ are real vectors in $\mathbb{R}^N$ and $\mu_{j,0}, \nu_{j,0} \in \mathbb{R}$. This use of $\mu_{j,0}$, $\nu_{j,0}$ terms generalizes the utility functions.

We now define the leader's utility matrix U and the follower's utility matrix V as the following,

$$U = \begin{pmatrix} \mu_{1,0} & \dots & \mu_{J,0} \\ \boldsymbol{\mu}_1 & \dots & \boldsymbol{\mu}_J \end{pmatrix}, V = \begin{pmatrix} \nu_{1,0} & \dots & \nu_{J,0} \\ \boldsymbol{\nu}_1 & \dots & \boldsymbol{\nu}_J \end{pmatrix}.$$

Then for a leader's strategy $\mathbf{x}$, the leader and follower's J utilities for the follower's J pure strategies are $U^{\mathrm{T}}\begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}$ and $V^{\mathrm{T}}\begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}$.

A Bayesian extension to the Stackelberg game allows multiple types of players, each with its own payoff matrix. We represent a Bayesian Stackelberg game with S follower types by a set of utility matrix pairs $(U^1, V^1), \dots, (U^S, V^S)$, each corresponding to a type. A type s has a prior probability p^s representing the likelihood of its occurrence. The leader commits to a mixed strategy without knowing the type of the follower she faces. The follower, however, knows his own type s, and plays the best response $j^s \in \{1, \dots, J\}$ according to his utility matrix V^s. A strategy profile in a Bayesian Stackelberg game is $\langle \mathbf{x}, \mathbf{j} \rangle$, a pair of leader's mixed strategy $\mathbf{x}$ and follower's response $\mathbf{j}$, where $\mathbf{j} = \langle j^1, \dots, j^S \rangle$ denotes a vector of the follower's responses for all types.

Type 1	*Target1*	*Target2*
Target1	1, -1	-1, 0
Target2	0, 1	1, -1

Type 2	*Target1*	*Target2*
Target1	1, -1	-1, 1
Target2	0, 1	1, -1

Figure 1: Payoff matrices of a Bayesian Stackelberg game.

The solution concept of interest is a *Strong Stackelberg Equilibrium* (SSE) [15], where the leader maximizes her expected utility assuming the follower chooses the best response and breaks ties in favor of the leader for each type. Formally, let $u(\mathbf{x}, \mathbf{j}) = \sum_{s=1}^{S} p^s((\boldsymbol{\mu}_{j^s}^s)^{\mathrm{T}}\mathbf{x} + \mu_{j^s,0}^s)$ denote the leader's expected utility, and $v^s(\mathbf{x}, j^s) = (\boldsymbol{\nu}_{j^s}^s)^{\mathrm{T}}\mathbf{x} + \nu_{j^s,0}^s$ denote the follower's expected utility for a type s. Then, $\langle \mathbf{x}^*, \mathbf{j}^* \rangle$ is an SSE if and only if,

$$\langle \mathbf{x}^*, \mathbf{j}^* \rangle = \arg\max_{\mathbf{x}, \mathbf{j}} \{u(\mathbf{x}, \mathbf{j}) | v^s(\mathbf{x}, j^s) \geq v^s(\mathbf{x}, j'), \forall j' \neq j^s\}.$$

As an example, which we will return to throughout the paper, consider a Bayesian Stackelberg game with two follower types, where type 1 appears with probability .84 and type 2 appears with probability .16. The leader (defender) chooses a probability distribution of allocating one resource to protect the two targets whereas the follower (attacker) chooses the best target to attack. We show the payoff matrices in Figure 1, where the leader is the row player and the follower is the column player. The utilities of the two types are identical except that a follower of type 2 gets a utility of 1 for attacking *Target2* successfully, whereas one of type 1 gets 0. The leader's strategy is a column vector $(x_1, x_2)^{\mathrm{T}}$ representing the probabilities of protecting the two targets. Given one resource, the strategy space of the leader is $x_1 + x_2 \leq 1, x_1 \geq 0, x_2 \geq 0$, i.e., $A = (1, 1), \mathbf{b} = 1$. The payoffs in Figure 1 can be represented by the following utility matrices,

$$U^1 = \begin{pmatrix} 0 & 0 \\ 1 & -1 \\ 0 & 1 \end{pmatrix}, V^1 = \begin{pmatrix} 0 & 0 \\ -1 & 0 \\ 1 & -1 \end{pmatrix};$$
$$U^2 = \begin{pmatrix} 0 & 0 \\ 1 & -1 \\ 0 & 1 \end{pmatrix}, V^2 = \begin{pmatrix} 0 & 0 \\ -1 & 1 \\ 1 & -1 \end{pmatrix}.$$

In terms of previous work, Bayesian Stackelberg games have been typically solved via tree search, where we assign one follower type to a pure strategy at each tree level [8]. For example, Figure 2 shows the search tree of the example game in Figure 1. We solve four linear programs, one for each leaf node. At each leaf node, the linear program provides an optimal leader strategy such that the follower's best response for every follower type is the chosen target at that leaf node, e.g., at the leftmost leaf node, the linear program finds the optimal leader strategy such that both type 1 and type 2 have a best response of attacking *Target1*. Comparing across leaf nodes, we obtain the overall optimal leader strategy [5]. In this case, the leaf node where type 1 is assigned to *Target1* and type 2 to *Target2* provides the overall optimal strategy.

Instead of solving an LP for all J^S leaf nodes, recent work uses a branch-and-bound technique to speed up the tree search [8]. The key to efficiency in branch-and-bound is obtaining tight upper and lower bounds for internal nodes, i.e., for nodes shown by circles in Figure 2, where subsets of follower types are assigned to particular targets. For example, in Figure 2, suppose the left subtree has been explored; now if at the rightmost internal node (where type 1 is assigned to *Target2*) we realize that the upper bound on solution quality is 0.5, we could prune the right subtree without even considering type 2. One possible way of obtaining upper bounds is by relaxing the integrality constraints in DOBSS MILP [12]. Unfortunately, when the integer variables in DOBSS are relaxed, the objective can be arbitrarily large, leading to meaningless upper bounds. HBGS [8] computes upper bounds by heuristically utilizing the solutions of smaller restricted games. However, the preprocessing involved in solving many small games can be expensive and the bounds computed using heuristics can again be loose.

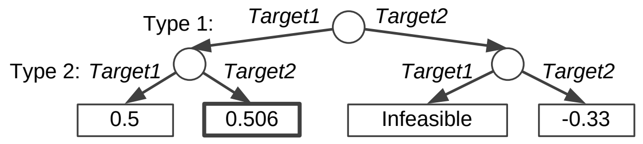


Figure 2: Example search tree of solving Bayesian games.

3. APPROACH

We present HUNTER (**H**andling **UN**cer**T**ainty **E**fficiently using **R**elaxation) based on the five key ideas mentioned in Section 1.

3.1 Algorithm Overview

To find the optimal leader's mixed strategy, HUNTER would conduct a best-first search in the search tree that results from assigning follower types to pure strategies, such as the search tree in Figure 2. Simply stated, HUNTER aims to search this space much more efficiently than HBGS [8]. As discussed earlier, efficiency gains are sought by obtaining tight upper bounds and lower bounds at internal nodes in the search tree (which corresponds to a partial assignment in which a subset of follower types are fixed). To that end, as illustrated in Figure 3, we use an upper bound LP within an internal search node. The LP returns an upper bound UB and a feasible solution $\mathbf{x}^*$, which is then evaluated by computing the follower best response, providing a lower bound LB. The solution returned by the upper bound LP is also utilized in choosing a new type s^* to create branches. To avoid having this upper bound LP itself become a bottleneck, it is solved efficiently using Bender's decomposition, which will be explained below.

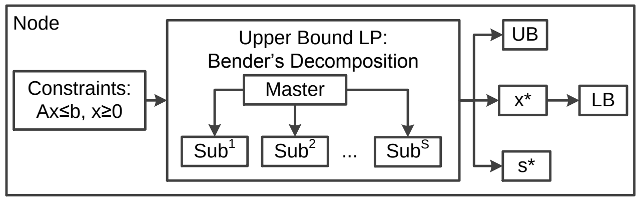


Figure 3: Steps of creating internal search nodes in HUNTER.

To understand HUNTER's behavior on a toy game instance, see Figure 4, which illustrates HUNTER's search tree in solving the example game from Figure 1 above. To start the best-first search, at the root node, no types are assigned any targets yet; we solve the upper bound LP with the initial strategy space $x_1 + x_2 \leq 1, x_1, x_2 \geq 0$ (Node 1). As a result, we obtain an upper bound of 0.560 and the optimal solution $x_1^* = 2/3, x_2^* = 1/3$. We evaluate the solution returned and obtain a lower bound of 0.506. Using HUNTER's heuristics, type 2 is then chosen to create branches by assigning it to *Target1* and *Target2* respectively. Next, we consider a child node (Node 2) in which type 2 is assigned to *Target1*, i.e., type 2's best response is to attack *Target1*. As a result, the follower's expected utility of choosing *Target1* must be higher than that of choosing *Target2*, i.e., $-x_1 + x_2 \geq x_1 - x_2$, simplified as $x_1 - x_2 \leq 0$. Thus, in Node 2, we impose an additional constraint $x_1 - x_2 \leq 0$ on the strategy space and obtain an upper bound of 0.5. Since its upper bound is lower than the current lower bound 0.506, this branch can be pruned out. Next we consider the other child node (Node 3) in which type 2 is assigned to *Target2*. This time we add constraint $-x_1 + x_2 \leq 0$ instead, and obtain an upper bound of 0.506. Since the upper bound coincides with the lower bound, we do not need to expand the node further. Moreover, since we have considered both *Target1* and *Target2* for type 2, we can terminate the algorithm and return 0.506 as the optimal solution value.

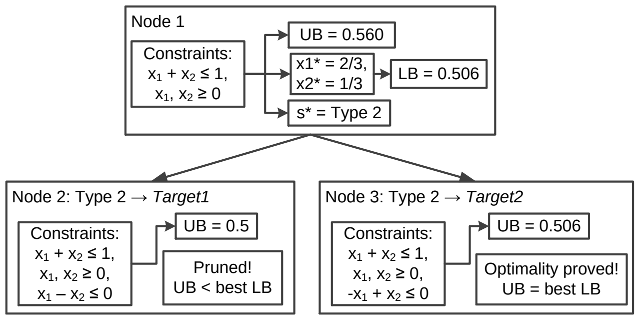


Figure 4: Example of internal nodes in HUNTER's search tree.

We now discuss HUNTER's behavior line-by-line (see Algorithm 1). We initialize the best-first search by creating the root node of the search tree with no assignment of types to targets and with the computation of the node's upper bound (Line 2 and 3). The initial lower bound is obtained by evaluating the solution returned by the upper bound LP (Line 4). We added the root node to a priority queue of open nodes which is internally sorted in a decreasing order of their upper bounds (Line 5). Each node contains information of the partial assignment, the feasible region of $\mathbf{x}$, the upper bound, and the Bender's cuts generated by the upper bound LP. At each iteration, we retrieve the node with the highest upper bound (Line 8), select a type s^* to assign pure strategies (Line 9), compute the upper bounds of the node's child nodes (Line 12 and 14), update the lower bound using the new solutions (Line 15), and enqueue child nodes with upper bound higher than the current lower bound (Line 16). As shown later, Bender's cuts at a parent node can be inherited by its children, speeding up the computation (Line 12).

Algorithm 1: HUNTER

```
1  Initialization;
2  [UB, $\mathbf{x}^*$, BendersCuts] = SolveUBLP($\phi$, $A\mathbf{x} \preceq \mathbf{b}$, $-\infty$);
3  root := ⟨ UB, $\mathbf{x}^*$, $A\mathbf{x} \preceq \mathbf{b}, x \succeq \mathbf{0}$, BendersCuts ⟩ ;
4  LB := Evaluate($\mathbf{x}^*$);
5  Enqueue(queue, root);
6  Best-first Search;
7  while not Empty(queue) do
8      node := pop(queue);
9      $s^*$ := PickType(node);
10     for $j := 1$ to $J$ do
11         NewConstraints := node.Constraints $\cup \{D_j^{s^*}\mathbf{x} + \mathbf{d}_j^{s^*} \succeq \mathbf{0}\}$ ;
12         [NewUB, $\mathbf{x}'$, NewBendersCuts] = SolveUBLP(node.BendersCuts,
           NewConstraints, LB) ;
13         if NewUB > LB then
14             child := ⟨ NewUB, $\mathbf{x}'$, NewConstraints, NewBendersCuts⟩ ;
15             LB := $\max$\{Evaluate($\mathbf{x}'$), LB\} ;
16             Enqueue(queue, child);
17         end
18     end
19 end
```

In the rest of the section, we will 1) present the upper bound LP, 2) show how to solve it using Bender's decomposition, and 3) verify the correctness of passing down Bender's cuts from parent to child nodes, 4) introduce the heuristic branching rule.

3.2 Upper Bound Linear Program

We derive a tractable linear relaxation of Bayesian Stackelberg games to provide an upper bound efficiently at each of HUNTER's internal nodes. For expository purpose, we focus on the root node of the search tree. Applying the results in disjunctive program [2], we first derive the convex hull for a single type. Then we show intersecting the convex hulls of all its types provides a tractable, polynomial-size relaxation of a Bayesian Stackelberg game.

3.2.1 Convex hull of a Single Type

Consider a Stackelberg game with a single follower type (U, V), the leader's optimal strategy $\mathbf{x}^*$ is the best among the optimal solutions of J LPs where each restricts the follower's best response to one pure strategy [5]. Hence we can represent the optimization problem as the following disjunctive program (i.e., a disjunction of "Multiple LPs" [5]),

$$\begin{array}{ll} \max\limits_{\mathbf{x},u} & u \\ s.t. & A\mathbf{x} \preceq \mathbf{b}, \mathbf{x} \succeq \mathbf{0} \\ & \bigvee\limits_{j=1}^{J} \begin{pmatrix} u \leq \boldsymbol{\mu}_j^{\mathrm{T}}\mathbf{x} + \mu_{j,0} \\ D_j\mathbf{x} + \mathbf{d}_j \preceq \mathbf{0} \end{pmatrix} \end{array} \tag{1}$$

where D_j and $\mathbf{d}_j$ are given by,

$$D_j = \begin{pmatrix} \boldsymbol{\nu}_1^{\mathrm{T}} - \boldsymbol{\nu}_j^{\mathrm{T}} \\ \vdots \\ \boldsymbol{\nu}_J^{\mathrm{T}} - \boldsymbol{\nu}_j^{\mathrm{T}} \end{pmatrix}, \mathbf{d}_j = \begin{pmatrix} \nu_{1,0} - \nu_{j,0} \\ \vdots \\ \nu_{J,0} - \nu_{j,0} \end{pmatrix}.$$

The feasible set of (1), denoted by H, is a union of J convex sets, each corresponding to a disjunctive term. Applying the results in [2], the closure of the convex hull of H, clconv$H$, is[1],

$$\left\{ \begin{array}{l|l} & \mathbf{x} = \sum\limits_{j=1}^{J} \boldsymbol{\chi}_j, \boldsymbol{\chi}_j \succeq \mathbf{0}, \forall j \\ u \in \mathbb{R} & u = \sum\limits_{j=1}^{J} \psi_j, \psi_j \geq 0, \forall j \\ \mathbf{x} \in \mathbb{R}^n & \sum\limits_{j=1}^{J} \theta_j = 1, \theta_j \geq 0, \forall j \\ & \begin{pmatrix} A & -\mathbf{b} & \mathbf{0} \\ D_j & \mathbf{d}_j & \mathbf{0} \\ -\boldsymbol{\mu}_j^{\mathrm{T}} & -\mu_{j,0} & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{\chi}_j \\ \theta_j \\ \psi_j \end{pmatrix} \preceq \mathbf{0}, \forall j \end{array} \right\}.$$

The intuition here is that the continuous variables $\boldsymbol{\theta}, \sum_{j=1}^{J} \theta_j = 1$ are used to create all possible convex combination of points in H. Furthermore, when $\theta_j \neq 0$, $\langle \frac{\boldsymbol{\chi}_j}{\theta_j}, \frac{\psi_j}{\theta_j} \rangle$ represents a point in the convex set defined by the j-th disjunctive term in the original problem (1). Finally, since all the extreme points of clconv$H$ belong to H, the disjunctive program (1) is equivalent to the linear program:

$$\max_{\mathbf{x},u} \{u | (\mathbf{x}, u) \in \text{clconv}H\}.$$

3.2.2 Tractable Relaxation

Building on the convex hulls of individual types, we now derive the relaxation of a Bayesian Stackelberg game with S types. We write this game with S types as the following disjunctive program,

$$\begin{array}{ll} \max\limits_{\mathbf{x},u^1,\dots,u^S} & \sum_{s=1}^{S} p^s u^s \\ s.t. & A\mathbf{x} \preceq \mathbf{b}, \mathbf{x} \succeq \mathbf{0} \\ & \bigwedge\limits_{s=1}^{S} \left[\bigvee\limits_{j=1}^{J} \begin{pmatrix} u^s \leq (\boldsymbol{\mu}_j^s)^{\mathrm{T}}\mathbf{x} + \mu_{j,0}^s \\ D_j^s\mathbf{x} + \mathbf{d}_j^s \preceq \mathbf{0} \end{pmatrix} \right] \end{array} \tag{2}$$

[1]To use the results in [2], we assume $u \geq 0$ for convenience. In the case where u can be negative, we can replace u by $u^+ - u^-$, with $u^+, u^- \geq 0$.

Returning to our toy example, the corresponding disjunctive program of the game in Figure 1 can be written as,

$$\begin{array}{ll} \max\limits_{x_1,x_2,u^1,u^2} & 0.84u^1 + 0.16u^2 \\ s.t. & x_1 + x_2 \leq 1, x_1, x_2 \geq 0 \\ & \begin{pmatrix} u^1 \leq x_1 \\ x_1 - 2x_2 \leq 0 \end{pmatrix} \bigvee \begin{pmatrix} u^1 \leq -x_1 + x_2 \\ -x_1 + 2x_2 \leq 0 \end{pmatrix} \\ & \begin{pmatrix} u^2 \leq x_1 \\ x_1 - x_2 \leq 0 \end{pmatrix} \bigvee \begin{pmatrix} u^2 \leq -x_1 + x_2 \\ -x_1 + x_2 \leq 0 \end{pmatrix} \end{array} \tag{3}$$

Denote the set of feasible points $(\mathbf{x}, u^1, \dots, u^S)$ of (2) by H^*. Unfortunately, to use the results of [2] here and create clconvH^*, we need to expand (2) to a disjunctive normal form, resulting in a linear program with an exponential number ($O(NJ^S)$) of variables. Instead, we now give a much more tractable, polynomial-size relaxation of (2). Denote the feasible set of each type s, $(\mathbf{x}, u^s)$ by H^s, and define $\widehat{H^*} := \{(\mathbf{x}, u^1, \dots, u^S) | (\mathbf{x}, u^s) \in \text{clconv}H^s, \forall s\}$. Then the following program is a relaxation of (2):

$$\max_{\mathbf{x},u^1,\dots,u^s} \left\{ \sum_{s=1}^{S} p^s u^s | (\mathbf{x}, u^s) \in \text{clconv}H^s, \forall s \right\} \tag{4}$$

Indeed, for any feasible point $(\mathbf{x}, u^1, \dots, u^S)$ in H^*, $(\mathbf{x}, u^s)$ must belong to H^s, implying that $(\mathbf{x}, u^s) \in \text{clconv}H^s$. Hence $H^* \subseteq \widehat{H^*}$, implying that optimizing over $\widehat{H^*}$ provides an upper bound on H^*. On the other hand, $\widehat{H^*}$ will in general have points not belonging to H^* and thus the relaxation can lead to an overestimation.

For example, consider the disjunctive program in (3). $(x_1 = \frac{2}{3}, x_2 = \frac{1}{3}, u^1 = \frac{2}{3}, u^2 = 0)$ does not belong to H^* since $-x_1 + x_2 \leq 0$ but $u^2 \not\leq -x_1 + x_2 = -\frac{1}{3}$. However the point belongs to $\widehat{H^*}$ because: i) $(x_1 = \frac{2}{3}, x_2 = \frac{1}{3}, u^1 = \frac{2}{3})$ belongs to $H^1 \subseteq$ clconvH^1; ii) $(x_1 = \frac{2}{3}, x_2 = \frac{1}{3}, u^2 = 0)$ belongs to clconvH^2, as it is the convex combination of two points in H^2: $(x_1 = \frac{1}{2}, x_2 = \frac{1}{2}, u^2 = \frac{1}{2})$ and $(x_1 = 1, x_2 = 0, u^2 = -1)$,

$$(\frac{2}{3}, \frac{1}{3}, 0) = \frac{2}{3} \times (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}) + \frac{1}{3} \times (1, 0, -1).$$

The upper bound LP (4) has $O(NJS)$ number of variables and constraints, and can be written as the following two-stage problem by explicitly representing clconvH^s:

$$\begin{array}{ll} \max\limits_{\mathbf{x}} & \sum\limits_{s=1}^{S} p^s u^s(\mathbf{x}) \\ s.t. & A\mathbf{x} \preceq \mathbf{b}, \mathbf{x} \succeq \mathbf{0} \end{array} \tag{5}$$

where $u^s(\mathbf{x})$ is defined to be the optimal value of,

$$\begin{array}{ll} \max\limits_{\boldsymbol{\chi}_j^s,\psi_j^s,\theta_j^s} & \sum\limits_{j=1}^{J} \psi_j^s, \;\; \psi_j^s \geq 0, \forall j \\ s.t. & \sum\limits_{j=1}^{S} \boldsymbol{\chi}_j^s = \mathbf{x}, \;\; \boldsymbol{\chi}_j^s \succeq \mathbf{0}, \forall j \\ & \sum\limits_{j=1}^{S} \theta_j^s = 1, \;\; \theta_j^s \geq 0, \forall j \\ & \begin{pmatrix} A & -\mathbf{b} & \mathbf{0} \\ D_j^s & \mathbf{d}_j^s & \mathbf{0} \\ -(\boldsymbol{\mu}_j^s)^{\mathrm{T}} & -\mu_{j,0}^s & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{\chi}_j^s \\ \theta_j^s \\ \psi_j^s \end{pmatrix} \preceq \mathbf{0}, \forall j \end{array} \tag{6}$$

Although written in two stages, the above formulation is in fact a single linear program, as both stages are maximization problems and combining the two stages will not produce any non-linear terms. We display formulations (5) and (6) in order to reveal the block structure for further speedup as explained below.

Note that so far, we have only derived the relaxation for the root node of HUNTER's search tree, without assigning any type to a pure strategy. This relaxation is also applied to other internal nodes in HUNTER's search tree. For example, if type s is assigned to pure strategy j, the leader's strategy space is further restricted by the addition of constraints of $D_j^s\mathbf{x} + \mathbf{d}_j^s \preceq \mathbf{0}$ to the original constraints $A\mathbf{x} \preceq \mathbf{b}, \mathbf{x} \succeq \mathbf{0}$. That is, we now have $A'\mathbf{x} \preceq \mathbf{b}', \mathbf{x} \succeq \mathbf{0}$, where $A' = \begin{pmatrix} D_j^s \\ A \end{pmatrix}$ and $\mathbf{b}' = \begin{pmatrix} -\mathbf{d}_j^s \\ \mathbf{b} \end{pmatrix}$.

3.3 Bender's Decomposition

Although much easier than solving a full Bayesian Stackelberg game, solving the upper bound LP can still be computationally challenging. Here we invoke the block structure of (4) we observed above, which partitioned it into (5) and (6), where, (5) is a master problem and (6) for $s = 1, \dots, S$ are S subproblems. This block structure allows us to solve the upper bound LP efficiently using multi-cut Bender's Decomposition [4]. Generally speaking, the computational difficulty of optimization problems increases significantly with the number of variables and constraints. Instead of considering all variables and constraints of a large problem simultaneously, Bender's decomposition partitions the problem into multiple smaller problems, which can then be solved in sequence. For completeness, we now briefly describe the technique.

In Bender's decomposition, the second-stage maximization problem (6) is replaced by its dual minimization counterpart, with dual variables $\boldsymbol{\lambda}_j^s, \boldsymbol{\pi}^s, \eta^s$ for $s = 1, \dots, S$:

$$\begin{aligned} u^s(\mathbf{x}) = & \min_{\boldsymbol{\lambda}_j^s \succeq \mathbf{0}, \boldsymbol{\pi}^s, \eta^s} (\boldsymbol{\pi}^s)^{\mathrm{T}}\mathbf{x} + \eta^s \\ s.t. & \begin{pmatrix} A^{\mathrm{T}} & (D_j^s)^{\mathrm{T}} & -\boldsymbol{\mu}_j^s \\ -\mathbf{b}^{\mathrm{T}} & (\mathbf{d}_j^s)^{\mathrm{T}} & -\mu_{i,0}^s \\ \mathbf{0}^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix} \boldsymbol{\lambda}_j^s + \begin{pmatrix} \boldsymbol{\pi}^s \\ \eta^s \\ -1 \end{pmatrix} \succeq \mathbf{0}, \forall i \end{aligned} \tag{7}$$

Since the feasible region of (7) is independent of $\mathbf{x}$, its optimal solution is reached at one of a finite number of extreme points (of the dual variables). Since $u^s(\mathbf{x})$ is the minimum of $(\boldsymbol{\pi}^s)^{\mathrm{T}}\mathbf{x} + \eta^s$ over all possible dual points, we know the following inequality must be true in the master problem,

$$u^s \leq (\boldsymbol{\pi}_k^s)^{\mathrm{T}}\mathbf{x} + \eta_k^s, \;\; k = 1, \dots, K \tag{8}$$

where $(\boldsymbol{\pi}_k^s, \eta_k^s), k = 1, \dots, K$ are all the dual extreme points. Constraints of type (8) for the master problem are called optimality cuts (infeasibility cuts, another type of constraint, turn out not to be relevant for our problem).

Since there are typically exponentially many extreme points for the dual formulation (7), generating all constraints of type (8) is not practical. Instead, Bender's decomposition starts by solving the master problem (5) with a subset of these constraints to find a candidate optimal solution $(\mathbf{x}^*, u^{1,*}, \dots, u^{S,*})$. It then solves S dual subproblems (7) to calculate $u^s(\mathbf{x}^*)$. If all the subproblems have $u^s(\mathbf{x}^*) = u^{s,*}$, the algorithm stops. Otherwise for those $u^s(\mathbf{x}^*) < u^{s,*}$, the corresponding constraints of type (8) are added to the master program for the next iteration.

3.4 Reusing Bender's Cuts

We can further speed up the upper bound LP computation at internal nodes of HUNTER's search tree by not creating all of the Bender's cuts from scratch; instead, we can reuse Bender's cuts from the parent node in its children. Suppose $u^s \leq (\boldsymbol{\pi}^s)^{\mathrm{T}}\mathbf{x} + \eta^s$ is a Bender's cut in the parent node. This means u^s cannot be greater than $(\boldsymbol{\pi}^s)^{\mathrm{T}}\mathbf{x} + \eta^s$ for any $\mathbf{x}$ in the feasible region of the parent node. Because a child node's feasible region is always more restricted than its parent's, we can conclude u^s cannot be greater than $(\boldsymbol{\pi}^s)^{\mathrm{T}}\mathbf{x} + \eta^s$ for any $\mathbf{x}$ in the child node's feasible region, i.e., $u^s \leq (\boldsymbol{\pi}^s)^{\mathrm{T}}\mathbf{x} + \eta^s$ must also be a valid cut for the child node.

3.5 Heuristic Branching Rules

Given an internal node in the search tree of HUNTER, we must decide on the type to branch on next, i.e., the type for which J child nodes will be created at the next lower level of the tree. As we show in Section 5 below, the type selected to branch on has a significant effect on efficiency. Intuitively, we should select a type whereby the upper bound at these children nodes will decrease most significantly. To that end, HUNTER chooses the type whose $\boldsymbol{\theta}^s$ returned by (6) violates the integrality constraint the most. Recall that $\boldsymbol{\theta}^s$ is used to generate convex combinations. The motivation here is that if all $\boldsymbol{\theta}^s$ returned by (6) are integer vectors, the solution of the upper bound LP (5) and (6) is a feasible point of the original problem (2), implying the relaxation already returns the optimal solution. More specifically, as suggested in [7], HUNTER chooses type s^* whose corresponding $\boldsymbol{\theta}^{s^*}$ has the maximum entropy, i.e., $s^* = \arg\max_s - \sum_{j=1}^{J} \theta_j^s \log \theta_j^s$.

4. CONTINUOUS UNCERTAINTY IN STACKELBERG GAMES

This section extends HUNTER to handle continuous uncertainty via the *sample average approximation* technique [1]. We first introduce the uncertain Stackelberg game model with continuously distributed uncertainty in leader's execution, follower's observation, and both players' utilities. Then we show the uncertain Stackelberg game model can be written as a two-stage mixed-integer stochastic program, to which existing convergence results of the sample average approximation technique apply. Finally, we show the sampled problems are equivalent to Bayesian Stackelberg games, and consequently could also be solved by HUNTER.

4.1 Uncertain Stackelberg Game Model

We consider the following types of uncertainty in Stackelberg games with known distributions. First, similar to [10], we assume there is uncertainty in both the leader and the follower's utilities U and V. Second, similar to [16], we assume the leader's execution and the follower's observation are noisy. In particular, we assume the executed strategy and observed strategy are linear perturbations of the intended strategy, i.e., when the leader commits to $\mathbf{x}$, the actual executed strategy is $\mathbf{y} = F^{\mathrm{T}}\mathbf{x} + \mathbf{f}$ and the observed strategy by the follower is $\mathbf{z} = G^{\mathrm{T}}\mathbf{x} + \mathbf{g}$, where $(F, \mathbf{f})$ and $(G, \mathbf{g})$ are uncertain. Here $\mathbf{f}$ and $\mathbf{g}$ are used to represent the execution and observation noise that is independent on $\mathbf{x}$. In addition, F and G are $N \times N$ matrices allowing us to model execution and observation noise that is linearly dependent on $\mathbf{x}$. Note that G and $\mathbf{g}$ can be dependent on F and $\mathbf{f}$. For example, we can represent an execution noise that is independent of $\mathbf{x}$ and follows a Gaussian distribution with $\mathbf{0}$ mean using $F = I_N$ and $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \Sigma)$, where I_N is the $N \times N$ identity matrix. We assume U, V, F, $\mathbf{f}$, G, and $\mathbf{g}$ are random variables, following some known continuous distributions. We use a vector $\xi = (U, V, F, \mathbf{f}, G, \mathbf{g})$ to represent a realization of the above inputs, and we use the notation $\xi(\omega)$ to represent the corresponding random variable.

We now show the uncertain Stackelberg game can be written as a two-stage mixed-integer stochastic program. Let $Q(\mathbf{x}, \xi)$ be the leader's utility for a strategy $\mathbf{x}$ and a realization ξ, assuming the follower chooses the best response. The first stage maximizes the expectation of leader's utility with respect to the joint probability distribution of $\xi(\omega)$, i.e., $\max_{\mathbf{x}}\{\mathbb{E}[Q(\mathbf{x},\xi(\omega))]|A\mathbf{x}\preceq\mathbf{b},\mathbf{x}\succeq\mathbf{0}\}$. The second stage computes $Q(\mathbf{x},\xi)$[2]:

$$\begin{array}{rrl} & Q(\mathbf{x},\xi)= & \boldsymbol{\mu}_{i^*}^{\mathrm{T}}(F^{\mathrm{T}}\mathbf{x}+\mathbf{f})+\mu_{i^*,0} \\ \text{where} & i^*= & \arg\max_{i=1}^{m}\boldsymbol{\nu}_i^{\mathrm{T}}(G^{\mathrm{T}}\mathbf{x}+\mathbf{g})+\nu_{i,0}. \end{array} \quad (9)$$

4.2 Sample Average Approximation

Sample average approximation is a popular solution technique for stochastic programs with continuously distributed uncertainty [1]. It can be applied to solving uncertain Stackelberg games as follows. First, a sample $\xi^1,\ldots,\xi^S$ of S realizations of the random vector $\xi(\omega)$ is generated. The expected value function $\mathbb{E}[Q(\mathbf{x},\xi(\omega))]$ can then be approximated by the sample average function $\frac{1}{S}\sum_{s=1}^{S}Q(\mathbf{x},\xi^s)$. The sampled problem is given by,

$$\max_{\mathbf{x}}\left\{\sum_{s=1}^{S}\frac{1}{S}Q(\mathbf{x},\xi^s)|A\mathbf{x}\preceq\mathbf{b},\mathbf{x}\succeq\mathbf{0}\right\}. \quad (10)$$

The sampled problem provides tighter and tighter statistical upper bound of the true problem with increasing number of samples [11]; the number of samples required to solve the true problem to a certain accuracy grows linearly in the dimension of $\mathbf{x}$ [1].

In the sampled problem, each sample ξ corresponds to a tuple $(U,V,F,\mathbf{f},G,\mathbf{g})$. The following proposition shows ξ is equivalent to some $\hat{\xi}$ where $\hat{F}=\hat{G}=I_N$ and $\hat{\mathbf{f}}=\hat{\mathbf{g}}=\mathbf{0}$, implying the sampled execution and observation noise can be handled by simply perturbing the utility matrices.

PROPOSITION 1. *For any leader's strategy* $\mathbf{x}$ *and follower's strategy* j*, both players get the same expected utilities in two noise realizations* $(U,V,F,\mathbf{f},G,\mathbf{g})$ *and* $(\hat{U},\hat{V},I_N,\mathbf{0},I_N,\mathbf{0})$*, where,*

$$\hat{U}=\begin{pmatrix}1 & \mathbf{f}^{\mathrm{T}}\\ \mathbf{0} & F\end{pmatrix}U,\hat{V}=\begin{pmatrix}1 & \mathbf{g}^{\mathrm{T}}\\ \mathbf{0} & G\end{pmatrix}V.$$

PROOF. We calculate both players' expected utility vectors for both noise realizations to establish the equivalence:

$$\hat{U}^{\mathrm{T}}\begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}=U^{\mathrm{T}}\begin{pmatrix}1 & \mathbf{0}^{\mathrm{T}}\\ \mathbf{f} & F^{\mathrm{T}}\end{pmatrix}\begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}=U^{\mathrm{T}}\begin{pmatrix}1\\ F^{\mathrm{T}}\mathbf{x}+\mathbf{f}\end{pmatrix}.$$

$$\hat{V}^{\mathrm{T}}\begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}=V^{\mathrm{T}}\begin{pmatrix}1 & \mathbf{0}^{\mathrm{T}}\\ \mathbf{g} & G^{\mathrm{T}}\end{pmatrix}\begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}=V^{\mathrm{T}}\begin{pmatrix}1\\ G^{\mathrm{T}}\mathbf{x}+\mathbf{g}\end{pmatrix}. \quad \square$$

A direct implication of Proposition 1 is that the sampled problem (10) and (9) is equivalent to a Bayesian Stackelberg game of S equally weighted types, with utility matrices $(\widehat{U^s},\widehat{V^s}), s=1,\ldots,S$. Hence, via sample average approximation, HUNTER could be used to solve Stackelberg games with continuous payoff, execution, and observation uncertainty.

4.3 A Unified Approach

Applying sample average approximation in Bayesian Stackelberg games with discrete follower types, we are able to handle both discrete and continuous uncertainty simultaneously using HUNTER. The idea is to replace each discrete follower type by a set of samples of the continuous distribution, converting the original Bayesian Stackelberg game to a larger one. The resulting problem could again be solved by HUNTER, providing a solution robust to both types of uncertainty.

[2](9) can be formulated as a mixed-integer linear program as in [12]

5. EXPERIMENTAL RESULTS

Since none of the existing algorithm can handle both discrete and continuous uncertainty in Stackelberg games, we conduct three sets of experiments considering i) only discrete uncertainty, ii) only continuous uncertainty, and iii) both types of uncertainty . The utility matrices were randomly generated from a uniform distribution between -100 and 100. Results were obtained on a standard 2.8GHz machine with 2GB main memory, and were averaged over 30 trials.

5.1 Handling Discrete Follower Types

For discrete uncertainty, we compare the runtime of HUNTER with DOBSS [12] and HBGS [8] (specifically, HBGS-F, the most efficient variant), the two best known algorithms for general Bayesian Stackelberg games. We compare these algorithms, varying the number of types and the number of pure strategies per player. The tests use a cutoff time of one hour for all three algorithms.

Figure 5(a) shows the performance of the three algorithms when the number of types increases. The games tested in this set have 5 pure strategies for each player. The x-axis shows the number of types, while the y-axis shows the runtime in seconds. As can be seen in Figure 5(a), HUNTER provides significant speed-up, of orders of magnitude over both HBGS and DOBSS[3](the line depicting HUNTER is almost touching the x-axis in Figure 5(a)). For example, we find that HUNTER can solve a Bayesian Stackelberg game with 50 types in 17.7 seconds on average, whereas neither HBGS nor DOBSS can solve an instance in an hour. Figure 5(b) shows the performance of the three algorithms when the number of pure strategies for each player increases. The games tested in this set have 10 types. The x-axis shows the number of pure strategies for each player, while the y-axis shows the runtime in seconds. HUNTER again provides significant speed-up over both HBGS and DOBSS. For example, HUNTER on average can solve a game with 13 pure strategies in 108.3 seconds, but HBGS and DOBSS take more than 30 minutes.

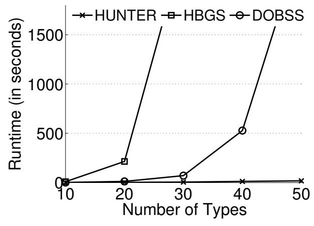
We now turn to analyzing the contributions of HUNTER's key components to its performance. First, we consider the runtime of HUNTER with two search heuristics, best-first (BFS) and depth-first (DFS), when the number of types is further increased. We set the pure strategies for each player to 5, and increase the number of types from 10 to 200. In Table 1, we summarize the average runtime and average number of nodes explored in the search process. As we can see, DFS is faster than BFS when the number of types is small, e.g., 10 types. However, BFS always explores significantly fewer number of nodes than DFS and is more efficient when the number types is large. For games with 200 types, the average runtime of BFS based HUNTER is 20 minutes, highlighting its scalability to a large number of types. Such scalability is achieved by efficient pruning – for a game with 200 types, HUNTER explores on average 5.3×10^3 nodes with BFS and 1.1×10^4 nodes with DFS, compared to a total of $5^{200}=6.2\times10^{139}$ possible leaf nodes.

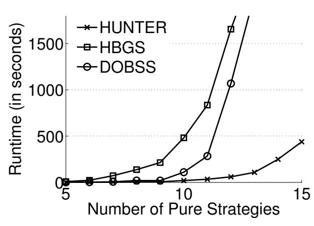
#Types	10	50	100	150	200
BFS Runtime (s)	5.7	17.7	178.4	405.1	1143.5
BFS #Nodes Explored	21	316	1596	2628	5328
DFS Runtime (s)	4.5	29.7	32.1	766.0	2323.5
DFS #Nodes Explored	33	617	3094	5468	11049

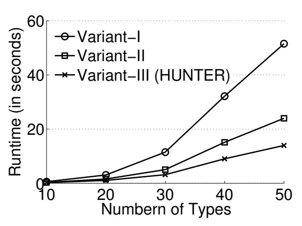
Table 1: Scalability of HUNTER to a large number of types

Second, we test the effectiveness of the two heuristics: inheritance of Bender's cuts from parent node to child nodes and the

[3]The runtime results of HBGS and DOBSS are inconsistent with the results in [8] because we use CPLEX 12 for solving mixed integer linear program instead of GLPK which is used in [8].


(a) Scaling up types.


(b) Scaling up pure strategies.


(c) Effectiveness of heuristics.

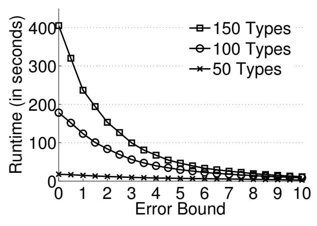

(d) Finding approximate solutions.

Figure 5: Experimental analysis of HUNTER and runtime comparison against HBGS, and DOBSS.

branching rule utilizing the solution returned by the upper bound LP. We fix the number of pure strategies for each agent to 5 and increase the number of types from 10 to 50. In Figure 5(c), we show the runtime results of three variants of HUNTER: i) Variant-I does not inherit Bender's cuts and chooses a random type to create branches; ii) Variant-II does not inherit Bender's cuts and uses the heuristic branching rule; iii) Variant-III (HUNTER) inherits Bender's cuts and uses the heuristic branching rule. The x-axis represents the number of types while the y-axis represents the runtime in seconds. As we can see, each individual heuristic helps speed up the algorithm significantly, showing their usefulness. For example, it takes 14.0 seconds to solve an instance of 50 types when both heuristics are enabled (Variant-III) compared to 51.5 seconds when neither of them is enabled (Variant-I).

Finally, we consider the performance of HUNTER in finding quality bounded approximate solutions. To this end, HUNTER is allowed to terminate once the difference between the upper bound and the lower bound decreases to η, a given error bound. The solution returned is therefore an approximate solution provably within η of the optimal solution. In this set of experiment, we test 30 games with 5 pure strategies for each player and 50, 100, and 150 types with varying error bound η from 0 to 10. As shown in Figure 5(d), HUNTER can effectively trade off solution quality for further speedup, indicating the effectiveness of its upper bound and lower bound heuristics. For example, for games with 100 types, HUNTER returns within 30 seconds a suboptimal solution at most 5 away from the optimal solution (the average optimal solution quality is 60.2). Compared to finding the global optimal solution in 178 seconds, HUNTER is able to achieve six-fold speedup by allowing at most 5 quality loss.

5.2 Handling Continuous Uncertainty

For continuous uncertainty, ideally we want to compare HUNTER with other algorithms that handle continuous execution and observation uncertainty in general Stackelberg games; but no such algorithm exists. Hence we restrict our investigation to the more restricted security games [9], so that two previous robust algorithms BRASS [13] and RECON [16] can be used in such a comparison. To introduce the uncertainty in these security games, we assume the defender's execution and the attacker's observation uncertainty follows independent uniform distributions. That is, for an intended defender strategy $\mathbf{x} = \langle x_1, \ldots, x_N \rangle$, where x_i represents the probability of protecting target i, we assume the maximum execution error associated with target i is α_i, and the actual executed strategy is $\mathbf{y} = \langle y_1, \ldots, y_N \rangle$, where y_i follows a uniform distribution between $x_i - \alpha_i$ and $x_i + \alpha_i$ for each i. Similarly, we assume the maximum observation error for target i is β_i, and the actual observed strategy is $\mathbf{z} = \langle z_1, \ldots, z_N \rangle$, where z_i follows a uniform distribution between $y_i - \beta_i$ and $y_i + \beta_i$ for each i. The definition of maximum error $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ is consistent with the definition in [16].

We use HUNTER with 20 samples and 100 samples to solve the problem above via sample average approximation as described in Section 4. For each setting, we repeat HUNTER 20 times with different sets of samples and report the best solution found (as shown below, HUNTER's competitors also try 20 settings and choose the best). Having generated a solution with 20 or 100 samples, evaluating its actual quality is difficult in the continuous uncertainty model – certainly any analytical evaluation is extremely difficult. Therefore, to provide an accurate estimation of the actual quality, we draw $10,000$ samples from the uncertainty distribution and evaluate the solution using these samples.

For comparison, we consider two existing robust solution methods BRASS [13] and RECON [16]. As experimentally tested in [10], when its parameter ϵ is chosen carefully, BRASS strategy is one of the top performing strategy under continuous payoff uncertainty. RECON assumes a maximum execution error $\boldsymbol{\alpha}$ and a maximum observation error $\boldsymbol{\beta}$, computing the risk-averse strategy for the defender that maximizes the worst-case performance over all possible noise realization. To provide a more meaningful comparison, we find solutions of BRASS / RECON repeatedly with multiple settings of parameters and report the best one. For BRASS, we test 20 ϵ settings, and for RECON, we set $\boldsymbol{\alpha} = \boldsymbol{\beta}$ and test 20 settings.

In our experiments, we test on 30 random generated security games with five targets and one resource. The maximum execution and observation error is set to $\boldsymbol{\alpha} = \boldsymbol{\beta} = 0.1$. The utilities in the game are drawn from a uniform distribution between -100 and $+100$. Nonetheless, the possible optimal solution quality lies in a much narrower range. Over the 30 instances we tested, the optimal solution quality we found by any algorithm varies between -26 and +17. In Table 2, we show the solution quality of HUNTER compared to BRASS and RECON respectively. #Wins shows the number of instances out of 30 where HUNTER returns a better solution than BRASS / RECON. Avg. Diff. shows the average gain of HUNTER over BRASS (or RECON), and the average solution quality of the corresponding algorithm (shown in the parentheses). Max. Diff. shows the maximum gain of HUNTER over BRASS (or RECON), and the solution quality of the corresponding instance and algorithm (shown in the parentheses). As we can see, HUNTER with 20 and 100 samples outperforms both BRASS and RECON on average. For example, RECON on average returns a solution with quality of -5.1, while even with 20 samples, the average gain HUNTER achieves over RECON is 0.6. The result is statistically significant with a paired t-test value of 8.9×10^{-6} and 1.0×10^{-3} for BRASS and RECON respectively. Indeed, when the number of samples used in HUNTER increases to 100, HUNTER is able to outperform both BRASS and RECON in every instance tested. Not only is the average difference in this case statistically significant, but the actual solution quality found by HUNTER– as shown by max difference – can be significantly better in practice than solutions found by BRASS and RECON.

	HUNTER-20 vs.		HUNTER-100 vs.	
	BRASS	RECON	BRASS	RECON
#Wins	27	24	30	30
Avg. Diff.	0.7(-5.2)	0.6(-5.1)	0.9(-5.2)	0.8(-5.1)
Max. Diff.	2.4(7.6)	4.0(-16.1)	3.31(7.6)	4.4(-16.1)

Table 2: Quality gain of HUNTER against BRASS and RECON under continuous execution and observation uncertainty.

5.3 Handling Both Types of Uncertainty

In our last experiment, we consider Stackelberg games with both discrete and continuous uncertainty. Since no previous algorithm can handle both, we only show the runtime results of HUNTER. We test on security games with five targets and one resource, and with multiple discrete follower types whose utilities are randomly generated. For each type, we use the same utility distribution and the same execution and observation uncertainty as in Section 5.2. Table 3 summarizes the runtime results of HUNTER for 3, 4, 5, 6 follower types, and 10, 20 samples per type. As we can see, HUNTER can efficiently handle both uncertainty simultaneously. For example, HUNTER spends less than 4 minutes on average to solve a problem with 5 follower types and 20 samples per type.

#Discrete Types	3	4	5	6
10 Samples	4.9	12.8	29.3	54.8
20 Samples	32.4	74.6	232.8	556.5

Table 3: Runtime results (in seconds) of HUNTER for handling both discrete and continuous uncertainty.

6. CONCLUSIONS

With increasing numbers of real-world security applications of leader-follower Stackelberg games, it is critical that we address uncertainty in such games, including discrete attacker types and continuous uncertainty such as the follower's observation noise, the leader's execution error, and both players' payoffs uncertainty. Previously, researchers have designed specialized sets of algorithms to handle these different types of uncertainty, e.g. algorithms for discrete follower types [8, 12] have been distinct from algorithms that handle continuous uncertainty [10]. However, in the real-world, a leader may face all of this uncertainty simultaneously, and thus we desire a single unified algorithm that handles all this uncertainty.

To that end, this paper provides a novel unified algorithm, called HUNTER, that handles discrete and continuous uncertainty by scaling up Bayesian Stackelberg games. The paper's contributions are in two parts. First it proposes the HUNTER algorithm. The novelty of HUNTER is in combining AI search techniques (e.g. best first search and heuristics) with techniques from Operations Research (e.g. disjunctive programming and Bender's decomposition). None of these are out-of-the-box techniques, however, and most of these techniques had not been applied earlier in the context of Stackelberg games even in isolation. Our novel contributions are in algorithmically specifying how these can be applied (and applied in conjunction with one another) within the context of Stackelberg games. The result is that HUNTER provides speedups of orders of magnitude over existing algorithms.

Second, we show that via sample average approximation, HUNTER handles continuously distributed uncertainty. While no algorithm other than HUNTER exists to handle such continuous uncertainty in general Stackelberg games, we find, even in restricted settings of security games, HUNTER performs better than competitors focusing on robust solutions [16, 13]. Finally, the paper illustrates HUNTER's unique ability to handle both discrete and continuous uncertainty simultaneously within a single problem.

7. ACKNOWLEDGEMENT

This research was supported by the United States Department of Homeland Security through the Center for Risk and Economic Analysis of Terrorism Events (CREATE). We thank Matthew P. Johnson for detailed comments.

8. REFERENCES

[1] S. Ahmed, A. Shapiro, and E. Shapiro. The sample average approximation method for stochastic programs with integer recourse. *SIAM Journal of Optimization*, 12:479–502, 2002.

[2] E. Balas. Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics*, 89(1-3):3 – 44, 1998.

[3] N. Basilico, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS*, 2009.

[4] J. R. Birge and F. V. Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34(3):384 – 392, 1988.

[5] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *ACM EC-06*, pages 82–90, 2006.

[6] J. P. Dickerson, G. I. Simari, V. S. Subrahmanian, and S. Kraus. A graph-theoretic approach to protect static and moving targets from adversaries. In *AAMAS*, 2010.

[7] A. Gilpin and T. Sandholm. Information-theoretic approaches to branching in search. *Discrete Optimization*, 8(2):147 – 159, 2011.

[8] M. Jain, C. Kiekintveld, and M. Tambe. Quality-bounded solutions for finite bayesian stackelberg games: Scaling up. In *AAMAS*, 2011.

[9] M. Jain, J. Tsai, J. Pita, C. Kiekintveld, S. Rathi, M. Tambe, and F. Ordóñez. Software Assistants for Randomized Patrol Planning for the LAX Airport Police and the Federal Air Marshals Service. *Interfaces*, 40:267–290, 2010.

[10] C. Kiekintveld, J. Marecki, and M. Tambe. Approximation methods for infinite Bayesian Stackelberg games: Modeling distributional payoff uncertainty. In *AAMAS*, 2011.

[11] W.-K. Mak, D. P. Morton, and R. K. Wood. Monte carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, 24(1-2):47 – 56, 1999.

[12] P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordóñez, and S. Kraus. Playing games with security: An efficient exact algorithm for Bayesian Stackelberg games. In *AAMAS*, 2008.

[13] J. Pita, M. Jain, F. Ordóñez, M. Tambe, S. Kraus, and R. Magori-cohen. Effective solutions for real-world Stackelberg games: When agents must deal with human uncertainties. In *AAMAS*, 2009.

[14] J. Pita, M. Tambe, C. Kiekintveld, S. Cullen, and E. Steigerwald. Guards - game theoretic security allocation on a national scale. In *AAMAS (Industry Track)*, 2011.

[15] B. von Stengel and S. Zamir. Leadership with commitment to mixed strategies. Technical Report LSE-CDAM-2004-01, CDAM Research Report, 2004.

[16] Z. Yin, M. Jain, M. Tambe, and F. Ordonez. Risk-averse strategies for security games with execution and observational uncertainty. In *AAAI*, 2011.

Multi-Objective Optimization for Security Games

Matthew Brown[1], Bo An[1], Christopher Kiekintveld[2], Fernando Ordóñez[3], Milind Tambe[1]
[1]University of Southern California, Los Angeles, CA, 90089
[2]University of Texas at El Paso, El Paso, TX, 79968
[3]Universidad de Chile, Santiago, Chile
[1]{mattheab,boa,tambe}@usc.edu, [2]cdkiekintveld@utep.edu, [3]fordon@dii.uchile.cl

ABSTRACT

The burgeoning area of security games has focused on real-world domains where security agencies protect critical infrastructure from a diverse set of adaptive adversaries. There are security domains where the payoffs for preventing the different types of adversaries may take different forms (seized money, reduced crime, saved lives, etc) which are not readily comparable. Thus, it can be difficult to know how to weigh the different payoffs when deciding on a security strategy. To address the challenges of these domains, we propose a fundamentally different solution concept, multi-objective security games (MOSG), which combines security games and multi-objective optimization. Instead of a single optimal solution, MOSGs have a set of Pareto optimal (non-dominated) solutions referred to as the Pareto frontier. The Pareto frontier can be generated by solving a sequence of constrained single-objective optimization problems (CSOP), where one objective is selected to be maximized while lower bounds are specified for the other objectives. Our contributions include: (i) an algorithm, Iterative ϵ-Constraints, for generating the sequence of CSOPs; (ii) an exact approach for solving an MILP formulation of a CSOP (which also applies to multi-objective optimization in more general Stackelberg games); (iii) heuristics that achieve speedup by exploiting the structure of security games to further constrain a CSOP; (iv) an approximate approach for solving an algorithmic formulation of a CSOP, increasing the scalability of our approach with quality guarantees. Additional contributions of this paper include proofs on the level of approximation and detailed experimental evaluation of the proposed approaches.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed artificial intelligence—*Intelligent agents*

General Terms

Algorithms, Performance, Security

Keywords

Game Theory, Security, Multi-objective Optimization

1. INTRODUCTION

Game theory is an increasingly important paradigm for modeling security domains which feature complex resource allocation [5, 2]. Security games, a special class of attacker-defender Stackelberg games, are at the heart of several major deployed decision-support applications. Such systems include ARMOR at LAX airport [8], IRIS deployed by the US Federal Air Marshals Service [8], GUARDS developed for the US Transportation Security Administration [1], and PROTECT used in the Port of Boston by the US Coast Guard [1].

In these applications, the defender is trying to maximize a single objective. However, there are domains where the defender has to consider multiple objectives simultaneously. For example, the Los Angeles Sheriff's Department (LASD) needs to protect the city's metro system from ticketless travelers, common criminals, and terrorists.[1] From the perspective of LASD, each one of these attacker types provides a unique threat (lost revenue, property theft, and loss of life). Given this diverse set of threats, selecting a security strategy is a significant challenge as no single strategy can minimize the threat for all attacker types. Thus, tradeoffs must be made and protecting more against one threat may increase the vulnerability to another threat. However, it is not clear how LASD should weigh these threats when determining the security strategy to use. One could attempt to establish methods for converting the different threats into a single metric. However, this process can become convoluted when attempting to compare abstract notions such as safety and security with concrete concepts such as ticket revenue.

Bayesian security games have been used to model domains where the defender is facing multiple attacker types. The threats posed by the different attacker types are weighted according to the relative likelihood of encountering that attacker type. There are three potential factors limiting the use of Bayesian security games: (1) the defender may not have information on the probability distribution over attacker types, (2) it may be impossible or undesirable to directly compare and combine the defender rewards of different security games, and (3) only one solution is given, hiding the trade-offs between the objectives from the end user.

Thus, for many domains, including the LASD metro system, we propose a new game model, multi-objective security games (MOSG), which combines game theory and multi-objective optimization. The threats posed by the attacker types are treated as different objective functions which are not aggregated, thus eliminating the need for a probability distribution over attacker types. Unlike Bayesian security games which have a single optimal solution, MOSGs have a set of Pareto optimal (non-dominated) solutions which is referred to as the Pareto frontier. By presenting the Pareto frontier to the end user, they are able to better understand the structure of their problem as well as the tradeoffs between different security strategies. As a result, end users are able to make a more informed decision on which strategy to enact. For instance, LASD

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

[1]http://sheriff.lacounty.gov

has provided explicit feedback that rather than having a single option handed to them, they would prefer to be presented with a set of alternative strategies from which they can make a final selection.

In this paper, we describe the new MOSG solution concept and provide a set of algorithms for computing Pareto optimal solutions for MOSGs. Our key contributions include (i) Iterative ϵ-Constraints, an algorithm for generating the Pareto frontier for MOSGs by producing and solving a sequence of constrained single-objective optimization problems (CSOP); (ii) an exact approach for solving a mixed-integer linear program (MILP) formulation of a CSOP (which also applies to multi-objective optimization in more general Stackelberg games); (iii) heuristics that exploit the structure of security games to speedup solving CSOPs; and (iv) an approximate approach for solving CSOPs, which greatly increases the scalability of our approach while maintaining quality guarantees. Additionally, we provide analysis of the complexity and completeness for all of our algorithms as well as experimental results.

2. MOTIVATING DOMAINS

As mentioned earlier, LASD must protect the Los Angeles metro system from ticketless travelers, criminals, and terrorists. Each type of perpetrator is distinct and presents a unique set of challenges. Thus, LASD may have different payoffs for preventing the various perpetrators. Targeting ticketless travelers will increase the revenue generated by the metro system as it will encourage passengers to purchase tickets. Pursuing criminals will reduce the amount of vandalism and property thefts, increasing the overall sense of passenger safety. Focusing on terrorists could help to prevent or mitigate the effect of a future terrorist attack, potentially saving lives. LASD has finite resources with which to protect all of the stations in the city. Thus, it is not possible to protect all stations against all perpetrators at all times. Therefore, strategic decisions must be made such as where to allocate security resources and for how long. These allocations should be determined by the amount of benefit they provide to LASD. However, if preventing different perpetrators provides different, incomparable benefits to LASD, it may be unclear how to decide on a strategy. In such situations, a multi-objective security game model could be of use, since the set of Pareto optimal solutions can explore the trade-offs between the different objectives. LASD can then select the solution they feel most comfortable with based on the information they have.

3. MULTI-OBJECTIVE SECURITY GAMES

A multi-objective security game is a multi-player game between a defender and n attackers.[2] The defender tries to prevent attacks by covering targets $T = \{t_1, t_2, \dots, t_{|T|}\}$ using m identical resources which can be distributed in a continuous fashion amongst the targets. The defender's strategy can be represented as a coverage vector $\mathbf{c} \in C$ where c_t is the amount of coverage placed on target t and represents the probability of the defender successfully preventing any attack on t [9]. $C = \{\langle c_t\rangle | 0 \leq c_t \leq 1, \sum_{t\in T} c_t \leq m\}$ is the defender's strategy space. The attacker i's mixed strategy $\mathbf{a}_i = \langle a_i^t\rangle$ is a vector where a_i^t is the probability of attacking t.

U defines the payoff structure for an MOSG, with U_i defining the payoffs for the security game played between the defender and attacker i. $U_i^{c,d}(t)$ is the defender's utility if t is chosen by attacker i and is fully covered by a defender resource. If t is not covered, the defender's penalty is $U_i^{u,d}(t)$. The attacker's utility is denoted similarly by $U_i^{c,a}(t)$ and $U_i^{u,a}(t)$. A property of security games is that $U_i^{c,d}(t) > U_i^{u,d}(t)$ and $U_i^{u,a}(t) > U_i^{c,a}(t)$ which means that placing more coverage on a target is always beneficial for the defender and disadvantageous for the attacker [9]. For a strategy profile $\langle \mathbf{c}, \mathbf{a}_i\rangle$ for the game between the defender and attacker i, the expected utilities for both agents are given by:

$$U_i^d(\mathbf{c}, \mathbf{a}_i) = \sum_{t\in T} a_i^t U_i^d(c_t, t), \quad U_i^a(\mathbf{c}, \mathbf{a}_i) = \sum_{t\in T} a_t U_i^a(c_t, t)$$

where $U_i^d(c_t, t) = c_t U_i^{c,d}(t) + (1 - c_t) U_i^{u,d}(t)$ and $U_i^a(c_t, t) = c_t U_i^{c,a}(t) + (1 - c_t) U_i^{u,d}(t)$ are the payoff received by the defender and attacker i, respectively, if target t is attacked and is covered with c_t resources.

The standard solution concept for a two-player Stackelberg game is Strong Stackelberg Equilibrium (SSE) [14], in which the defender selects an optimal strategy based on the assumption that the attacker will choose an optimal response, breaking ties in favor of the defender. We denote $U_i^d(\mathbf{c})$ and $U_i^a(\mathbf{c})$ as the payoff received by the defender and attacker i, respectively, when the defender uses the coverage vector $\mathbf{c}$ and attacker i attacks the best target while breaking ties in favor of the defender.

With multiple attackers, the defender's utility (objective) space can be represented as a vector $U^d(\mathbf{c}) = \langle U_i^d(\mathbf{c})\rangle$. An MOSG defines a multi-objective optimization problem:

$$\max_{\mathbf{c}\in C} \left(U_1^d(\mathbf{c}), \dots, U_n^d(\mathbf{c})\right)$$

Solving such multi-objective optimization problems is a fundamentally different task than solving a single-objective optimization problem. With multiple objectives functions there exist tradeoffs between the different objectives such that increasing the value of one objective decreases the value of at least one other objective. Thus for multi-objective optimization, the traditional concept of optimality is replaced by Pareto optimality.

DEFINITION 1. *(Dominance). A coverage vector $\mathbf{c} \in C$ is said to* ***dominate*** $\mathbf{c}' \in C$ *if $U_i^d(\mathbf{c}) \geq U_i^d(\mathbf{c}')$ for all $i = 1, \dots, n$ and $U_i^d(\mathbf{c}) > U_i^d(\mathbf{c}')$ for at least one index i.*

DEFINITION 2. *(Pareto Optimality) A coverage vector $\mathbf{c} \in C$ is* ***Pareto optimal*** *if there is no other $\mathbf{c}' \in C$ that dominates $\mathbf{c}$. The set of non-dominated coverage vectors is called* ***Pareto optimal solutions*** *C^* and the corresponding set of objective vectors $\Omega = \{U^d(\mathbf{c}) | \mathbf{c} \in C^*\}$ is called the* ***Pareto frontier***.

This paper gives algorithms to find Pareto optimal solutions in MOSGs. If there are a finite number of Pareto optimal solutions, it is preferable to generate all of them for the end-user. If there are an infinite number of Pareto optimal solutions, it is impossible to generate all the Pareto optimal solutions. In this case, it is necessary to generate a subset of Pareto optimal solutions that can approximate the true Pareto frontier with quality guarantees. The methods we present in this paper are a starting point for further analysis and additional preference elicitation from end users, all of which depends on fast approaches for generating the Pareto frontier.

4. RELATED WORK

MOSGs build on security games and multi-objective optimization. We have already reviewed (in Section 1) the relationship of MOSGs to previous work in security games and in particular Bayesian security games. In this section, we primarily review the research on multi-objective optimization. There are three representative approaches for generating the Pareto frontier in multi-objective optimization problems. Weighted summation [4], where

[2]The defender does actually face multiple attackers of different types, however, these attackers are not coordinated and hence the problem we address is different than in [10].

the objective functions are assigned weights and aggregated, producing a single Pareto optimal solution. The Pareto frontier can then be explored by sampling different weights. Another approach is multi-objective evolutionary algorithms (MOEA) [6]. Evolutionary approaches such as NSGA-II [7] are capable of generating multiple approximate solutions in each iteration. However, due to their stochastic nature, both weighted summation and MOEA cannot bound the level of approximation for the generated Pareto frontier. This lack of solution quality guarantees is unacceptable for security domains on which we are focused.

The third approach is the ϵ-constraint method in which the Pareto frontier is generated by solving a sequence of CSOPs. One objective is selected as the primary objective to be maximized while lower bound constraints are added for the secondary objectives. The original ϵ-constraint method [4] discretizes the objective space and solves a CSOP for each grid point. This approach is computationally expensive since it exhaustively searches the objective space of secondary objectives. There has been work to improve upon the original ϵ-constraint method. [11] proposes an adaptive technique for constraint variation that leverages information from solutions of previous CSOPs. However, this method requires solving $\mathcal{O}(k^{n-1})$ CSOPs, where k is the number of solutions in the Pareto frontier. Another approach, the augmented ϵ-constraint method [12] reduces computation by using infeasibility information from previous CSOPs. However, this approach only returns a predefined number of points and thus cannot bound the level of approximation for the Pareto frontier. Our approach for solving an MOSG builds upon the basic idea of the ϵ-constraint method. Security domains demand *both* efficiency as well as quality guarantees when providing decision support. Our approach only needs to solve $\mathcal{O}(nk)$ CSOPs and can provide approximation bounds.

5. ITERATIVE ϵ-CONSTRAINTS

The ϵ-constraint method formulates a CSOP for a given set of constraints $\mathbf{b}$, producing a single Pareto optimal solution. The Pareto frontier is then generated by solving multiple CSOPs produced by modifying the constraints in $\mathbf{b}$. This section presents Iterative ϵ-Constraints, an algorithm for systematically generating a sequence of CSOPs for an MOSG. These CSOPs can then be passed to a solver Φ to return solutions to the MOSG. The next two sections present 1) an exact MILP approach (Section 6) which can guarantee that each solution is Pareto optimal and 2) a faster approximate approach (Section 7) for solving CSOPs.

5.1 Algorithm for Generating CSOPs

Iterative ϵ-Constraints uses the following four key ideas: 1) The Pareto frontier for an MOSG can be found by solving a sequence of CSOPs. For each CSOP, $U_1^d(\mathbf{c})$ is selected as the primary objective, which will be maximized. Lower bound constraints $\mathbf{b}$ are then added for the secondary objectives $U_2^d(\mathbf{c}), \ldots, U_n^d(\mathbf{c})$. 2) The sequence of CSOPs are iteratively generated by exploiting previous Pareto optimal solutions and applying Pareto dominance. 3) It is possible for a CSOP to have multiple coverage vectors $\mathbf{c}$ that maximize $U_1^d(\mathbf{c})$ and satisfy $\mathbf{b}$. Thus, lexicographic maximization is used to ensure that CSOP solver Φ only returns Pareto optimal solutions. 4) It may be impractical (even impossible) to generate all Pareto optimal points if the frontier contains a large number of points, e.g., the frontier is continuous. Therefore, a parameter ϵ is used to discretize the objective space, trading off solution efficiency versus the degree of approximation in the generated Pareto frontier.

We now present a simple MOSG example with two objectives and $\epsilon = 5$. Figure 5.1 shows the objective space for the problem as well as several points representing the objective vectors for

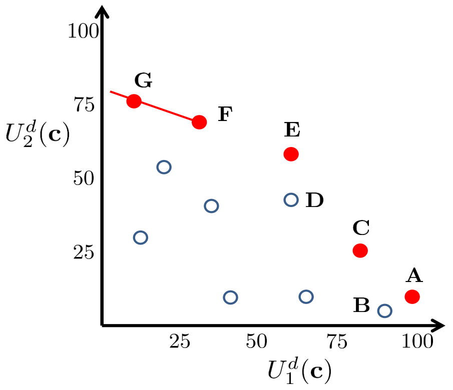


Figure 1: Pareto Frontier for a Bi-Objective MOSG

different defender coverage vectors. In this problem, U_1^d will be maximized while b_2 constrains U_2^d. The initial CSOP is unconstrained (i.e., $b_2 = -\infty$), thus the solver Φ will maximize U_1^d and return solution A = (100,10). Based on this result, we know that any point $\mathbf{v} = \{v_1, v_2\}$ (e.g., B) is not Pareto optimal if $v_2 < 10$, as it would be dominated by A. We then generate a new CSOP, updating the bound to $b_2 = 10 + \epsilon$. Solving this CSOP with Φ produces solution C=(80, 25) which can be used to generate another CSOP with $b_2 = 25 + \epsilon$. Both D=(60,40) and E=(60,60) satisfy b_2 but only E is Pareto optimal. Lexicographic maximization ensures that only E is returned and dominated solutions are avoided (details in Section 6). The method then updates $b_2 = 60 + \epsilon$ and Φ returns F=(30,70), which is part of a continuous region of the Pareto frontier from $U_2^d = 70$ to $U_2^d = 78$. The parameter ϵ causes the method to select a subset of the Pareto optimal points in this continuous region. In particular this example returns G=(10,75) and in the next iteration ($b_2 = 80$) finds that the CSOP is infeasible and terminates. The algorithm returns a Pareto frontier of A, C, E, F, and G.

Algorithm 1 systematically updates a set of lower bound constraints $\mathbf{b}$ to generate the sequence of CSOPs. Each time we solve a CSOP, a portion of the $n-1$ dimensional space formed by the secondary objectives is marked as searched with the rest divided into $n-1$ subregions (by updating $\mathbf{b}$ for each secondary objective). These $n-1$ subregions are then recursively searched by solving CSOPs with updated bounds. This systematic search forms a branch and bound search tree with a branching factor of $n-1$. As the depth of the tree increases, the CSOPs are more constrained, eventually becoming infeasible. If a CSOP is found to be infeasible, no child CSOPs are generated because they are guaranteed to be infeasible as well. The algorithm terminates when the entire secondary objective space has been searched.

Algorithm 1: Iterative-ϵ-Constraints($\mathbf{b} = \{b_2, \ldots, b_n\}$)

```
1  if b ∉ $previousBoundsList$ then
2      append($previousBoundsList$, b) ;
3      c ← $\Phi$(b) ;
4      if c is a feasible solution then
5          v ← $\{U_1^d(\mathbf{c}), \ldots, U_n^d(\mathbf{c})\}$;
6          for $2 \le i \le n$ do
7              b′ ← b;
8              $b'_i \leftarrow v_i + \epsilon$ ;
9              if b′ $\not\geq$ s, $\forall$s ∈ $infeasibleBoundsList$ then
10                 Iterative-$\epsilon$-Constraints(b′) ;
11     else append($infeasibleBoundsList$, b) ;
```

Two modifications are made to improve the efficiency of the algorithm. 1) Prevent redundant computation resulting from multiple nodes having an identical set of lower bound constraints by recording the lower bound constraints for all previous CSOPs in a list called $previousBoundsList$. 2) Prevent the solving of CSOPs which are known to be infeasible based on previous CSOPs by recording the lower bound constraints for all infeasible CSOPs in a list called $infeasibleBoundsList$.

5.2 Approximation Analysis

Assume the full Pareto frontier is Ω and the objective space of the solutions found by the Iterative ϵ-Constraints method is Ω_ϵ.

THEOREM 3. *Solutions in Ω_ϵ are non-dominated, i.e., $\Omega_\epsilon \subseteq \Omega$.*

PROOF. Let $\mathbf{c}^*$ be the coverage vector such that $U^d(\mathbf{c}^*) \in \Omega_\epsilon$ and assume that it is dominated by a solution from a coverage vector $\bar{\mathbf{c}}$. That means $U_i^d(\bar{\mathbf{c}}) \geq U_i^d(\mathbf{c}^*)$ for all $i = 1, \ldots, n$ and for some j, $U_j^d(\bar{\mathbf{c}}) > U_j^d(\mathbf{c}^*)$. This means that $\bar{\mathbf{c}}$ was a feasible solution for the CSOP for which $\mathbf{c}^*$ was found to be optimal. Furthermore, the first time the objectives differ, the solution $\bar{\mathbf{c}}$ is better and should have been selected in the lexicographic maximization process. Therefore $\mathbf{c}^* \notin \Omega_\epsilon$ which is a contradiction. □

Given the approximation introduced by ϵ, one immediate question is to characterize the efficiency loss. Here we define a bound to measure the largest efficiency loss:

$$\rho(\epsilon) = \max_{\mathbf{v} \in \Omega \setminus \Omega_\epsilon} \min_{\mathbf{v}' \in \Omega_\epsilon} \max_{1 \leq i \leq n} (v_i - v'_i)$$

This approximation measure is widely used in multi-objective optimization (e.g. [3]). It computes the maximum distance between any point $\mathbf{v} \in \Omega \setminus \Omega_\epsilon$ on the frontier to its "closest" point $\mathbf{v}' \in \Omega_\epsilon$ computed by our algorithm. The distance between two points is the maximum difference of different objectives.

THEOREM 4. $\rho(\epsilon) \leq \epsilon$.

PROOF. It suffices to prove this theorem by showing that for any $\mathbf{v} \in \Omega \setminus \Omega_\epsilon$, there is at least one point $\mathbf{v}' \in \Omega_\epsilon$ such that $v'_1 \geq v_1$ and $v'_i \geq v_i - \epsilon$ for $i > 1$.

Algorithm 2 recreates the sequence of CSOP problems generated by Iterative ϵ-Constraints but ensuring that the bound $\mathbf{b} \leq \mathbf{v}$ throughout. Since Algorithm 2 terminates when we do not update $\mathbf{b}$, this means that $v'_i + \epsilon > v_i$ for all $i > 1$. Summarizing, the final solution $\mathbf{b}$ and $\mathbf{v}' = U^d(\Phi(\mathbf{b}))$ satisfy $\mathbf{b} \leq \mathbf{v}$ and $v'_i > v_i - \epsilon$ for all $i > 1$. Since $\mathbf{v}$ is feasible for the CSOP with bound $\mathbf{b}$, but $\Phi(\mathbf{b}) = \mathbf{v}' \neq \mathbf{v}$ then $v'_1 \geq v_1$. □

Given Theorem 4, the maximum distance for every objective between any missed Pareto optimal point and the closest computed Pareto optimal point is bounded by ϵ. Therefore, as ϵ approaches 0, the generated Pareto frontier approaches the complete Pareto frontier in the measure $\rho(\epsilon)$. For example if there are k discrete solutions in the Pareto frontier and the smallest distance between any two is δ then setting $\epsilon = \delta/2$ will make $\Omega_\epsilon = \Omega$. In this case, since each solution corresponds to a non-leaf node in our search tree, the number of leaf nodes is no more than $(n-1)k$. Thus our algorithm will solve at most $\mathcal{O}(nk)$ CSOPs.

6. MILP APPROACH

In Section 5, we introduced a high level search algorithm for generating the Pareto frontier by producing a sequence of CSOPs. In this section we present an exact approach for defining and solving a mixed-integer linear program (MILP) formulation of a CSOP for MOSGs. We then go on to show how heuristics that exploit the structure and properties of security games can be used to improve the efficiency of our MILP formulation.

Algorithm 2: For $\mathbf{v} \in \Omega \setminus \Omega_\epsilon$, find $\mathbf{v}' \in \Omega_\epsilon$ satisfying $v'_1 \geq v_1$ and $v'_i \geq v_i - \epsilon$ for $i > 1$

```
1 Let $\mathbf{b}$ be the constraints in the root node, i.e., $b_i = -\infty$ for $i > 1$ ;
2 repeat
3     $\mathbf{c} \leftarrow \Phi(\mathbf{b}), \mathbf{v}' \leftarrow U^d(\mathbf{c}), \mathbf{b}' \leftarrow \mathbf{b}$;
4     for each objective $i > 1$ do
5         if $v'_i + \epsilon \leq v_i$ then
6             $b_i \leftarrow v'_i + \epsilon$ ;
7             break;
8 until $\mathbf{b} = \mathbf{b}'$;
9 return $\Phi(\mathbf{b})$ ;
```

$$\begin{array}{rcr}
 & \max \quad d_\lambda & (1)\\
1 \leq j \leq n, \forall t \in T: & d_j - U_j^d(c_t, t) \leq M(1 - a_j^t) & (2)\\
1 \leq j \leq n, \forall t \in T: & 0 \leq k_j - U_j^a(c_t, t) \leq M(1 - a_j^t) & (3)\\
1 \leq j < \lambda: & d_j = d_j^* & (4)\\
\lambda < j \leq n: & d_j \geq b_j & (5)\\
1 \leq j \leq n, \forall t \in T: & a_j^t \in \{0, 1\} & (6)\\
\forall j \in A: & \sum_{t \in T} a_j^t = 1 & (7)\\
\forall t \in T: & 0 \leq c_t \leq 1 & (8)\\
 & \sum_{t \in T} c_t \leq m & (9)
\end{array}$$

Figure 2: Lexicographic MILP Formulation for a CSOP

6.1 Exact MILP Method

As stated in Section 5, to ensure Pareto optimality of solutions lexicographic maximization is required to sequentially maximizing all the objective functions. Thus, for each CSOP we must solve n MILPs in the worst case where each MILP is used to maximize one objective. For the λ^{th} MILP in the sequence, the objective is to maximize the variable d_λ, which represents the defender's payoff for security game λ. This MILP is constrained by having to maintain the previously maximized values d_j^* for $1 \leq j < \lambda$ as well as satisfy lower bound constraints b_k for $\lambda < k \leq n$.

We present our MILP formulation for a CSOP for MOSGs in Figure 2. This is similar to the MILP formulations for security games presented in [9] and elsewhere with the exceptions of Equations (4) and (5). Equation (1) is the objective function, which maximizes the defender's payoff for objective λ, d_λ. Equation (2) defines the defender's payoff. Equation (3) defines the optimal response for attacker j. Equation (4) constrains the feasible region to solutions that maintain the values of objectives maximized in previous iterations of lexicographic maximization. Equation (5) guarantees that the lower bound constraints in $\mathbf{b}$ will be satisfied for all objectives which have yet to be optimized.

If a mixed strategy is optimal for the attacker, then so are all the pure strategies in the support of that mixed strategy. Thus, we only consider the pure strategies of the attacker [13]. Equations (6) and (7) constrain attackers to pure strategies that attack a single target. Equations (8) and (9) specify the feasible defender strategy space.

Once the MILP has been formulated, it can be solved using an optimization software package such as CPLEX. It is possible to increase the efficiency of the MILP formulation by using heuristics to constrain the decision variables. A simple example of a general heuristic which can be used to achieve speedup is placing an upper bound on the defender's payoff for the primary objective. Assume d_1 is the defender's payoff for the primary objective in the parent CSOP and d'_1 is the defender's payoff for the primary objective in

Variable	Definition	Dimension
λ	Current Objective	$-$
m	Number of Defender Resources	$-$
n	Number of Attacker Types	$-$
Z	Huge Positive Constant	$-$
T	Set of Targets	$\lvert T\rvert$
$\mathbf{a}$	Attacker Coverage a_j^t	$n \times \lvert T\rvert$
$\mathbf{b}$	Objective Bounds b_j	$(n-1) \times 1$
$\mathbf{c}$	Defender Coverage c_t	$\lvert T\rvert \times 1$
$\mathbf{d}$	Defender Payoff d_j	$n \times 1$
$\mathbf{d}^*$	Maximized Defender Payoff d_j^*	$n \times 1$
$\mathbf{k}$	Attacker Payoff k_j	$n \times 1$
U^d	Defender Payoff Structure $U_j^d(c_t, t)$	$n \times \lvert T\rvert$
U^a	Attacker Payoff Structure $U_j^a(c_t, t)$	$n \times \lvert T\rvert$

Figure 3: MILP Formulation Definitions

the child CSOP. As each CSOP is a maximization problem, it must hold that $d_1 \geq d'_1$ because the child CSOP is more constrained than the parent CSOP. Thus, the value of d_1 can be passed to the child CSOP to be used as an upper bound on the objective function.

As noted earlier, this MILP is a slight variation of the optimization problem formulated in [9] for security games. The same variations can be made to more generic Stackelberg games, such as those used for DOBSS [13], giving a formulation for multi-objective Stackelberg games in general.

6.2 Exploiting Game Structures

In addition to placing bounds on the defender payoff, it is possible to constrain the defender coverage in order to improve the efficiency of our MILP formulation. Thus, we introduce an approach for translating constraints on defender payoff into constraints on defender coverage. This approach, ORIGAMI-M, achieves this translation by computing the minimum coverage needed to satisfy a set of lower bound constraints $\mathbf{b}$ such that $U_i^d(\mathbf{c}) \geq b_i$ for $1 \leq i \leq n$. This minimum coverage is then added to the MILP in Figure 2 as constraints on the variable $\mathbf{c}$, reducing the feasible region and leading to significant speedup as verified in experiments.

ORIGAMI-M is a modified version of the ORIGAMI algorithm [9] and borrows many of its key concepts. At a high level, ORIGAMI-M starts off with an empty defender coverage vector $\mathbf{c}$, a set of lower bound constraints $\mathbf{b}$, and m defender resources. We try to compute a coverage $\mathbf{c}$ which uses the minimum defender resources to satisfy constraints $\mathbf{b}$. If a constraint b_i is violated, i.e., $U_i^d(\mathbf{c}) < b_i$, ORIGAMI-M updates $\mathbf{c}$ by computing the minimum additional coverage necessary to satisfy b_i. Since we focus on satisfying the constraint on one objective at a time, the constraints for objectives that were satisfied in previous iterations may become unsatisfied again. The reason is that additional coverage may be added to the target that was attacked by this attacker type, causing it to become less attractive relative to other alternatives for the attacker, and possibly reducing the defender's payoff by changing the target that is attacked. Therefore, the constraints in $\mathbf{b}$ must be checked repeatedly until quiescence (no chances are made to $\mathbf{c}$ for any b_i). If all m resources are exhausted before $\mathbf{b}$ is satisfied, then the CSOP is infeasible.

The process for calculating minimum coverage for a single constraint b_i is built on two properties of security games [9]: (1) the attacker chooses the optimal target; (2) the attacker breaks ties in favor of the defender. The set of optimal targets for attacker i for coverage $\mathbf{c}$ is referred to as the attack set, $\Gamma_i(\mathbf{c})$. Accordingly, adding coverage on target $t \notin \Gamma_i$ does not affect the attacker i's strategy or payoff. Thus, if $\mathbf{c}$ does not satisfy b_i, we only consider adding coverage to targets in Γ_i. Γ_i can be expanded by increasing coverage such that the payoff for each target in Γ_i is equivalent to the payoff for the next most optimal target. Adding an additional target to the attack set cannot hurt the defender since the defender receives the optimal payoff among targets in the attack set.

Algorithm 3: ORIGAMI-M($\mathbf{b}$)

```
1  c ← empty coverage vector ;
2  while $b_i > U_i^d(\mathbf{c})$ for some bound $b_i$ do
3      sort targets $T$ in decreasing order of value by $U_i^a(c_t, t)$;
4      $left \leftarrow m - \sum_{t \in T} c_t$, $next \leftarrow 2$;
5      while $next \leq |T|$ do
6          $addedCov[t] \leftarrow$ empty coverage vector;
7          if $\max_{1 \leq t < next} U_i^{c,a}(t) > U_i^a(c_{next}, t_{next})$ then
8              $x \leftarrow \max_{1 \leq t < next} U_i^{c,a}(t)$;
9              $noninducibleNextTarget \leftarrow true$;
10         else
11             $x \leftarrow U_i^a(c_{next}, t_{next})$;
12         for $1 \leq t < next$ do
13             $addedCov[t] \leftarrow \frac{x - U_i^{u,a}(t)}{U_i^{c,a}(t) - U_i^{u,a}(t)} - c_t$;
14         if $\sum_{t \in T} addedCov[t] > left$ then
15             $resourcesExceeded \leftarrow true$;
16             $ratio[t] \leftarrow \frac{1}{U_i^{u,a}(t) - U_i^{c,a}(t)}, \forall 1 \leq t < next$;
17             $addedCov[t] = \frac{ratio[t] \cdot left}{\sum_{1 \leq t \leq next} ratio[t]}, \forall 1 \leq t < next$;
18         if $U_i^d(\mathbf{c} + addedCov) \geq b_i$ then
19             $\mathbf{c}' \leftarrow$ MIN-COV($i, \mathbf{c}, \mathbf{b}$);
20             if $\mathbf{c}' \neq null$ then
21                 $\mathbf{c} \leftarrow \mathbf{c}'$
22             break;
23         else if resourcesExceeded ∨ noninducibleNextTarget then
24             return $infeasible$;
25         else
26             $c_t \mathrel{+}= addedCov[t], \forall t \in T$;
27             $left \mathrel{-}= \sum_{t \in T} addedCov[t]$;
28             $next$++;
29     if $next = |T| + 1$ then
30         if $left > 0$ then
31             $\mathbf{c} \leftarrow$ MIN-COV($i, \mathbf{c}, \mathbf{b}$);
32             if $\mathbf{c} = null$ then
33                 return $infeasible$;
34         else
35             return $infeasible$;
36 return c ;
```

The idea for ORIGAMI-M is to expand the attack set Γ_i until b_i is satisfied. The order in which the targets are added to Γ_i is by decreasing value of $U_i^a(c_t, t)$. Sorting these values, so that $U_i^a(c_1, t_1) \geq U_i^a(c_2, t_2) \geq \cdots \geq U_i^a(c_{|T|}, t_{|T|})$, we have that $\Gamma_i(\mathbf{c})$ starts only with target t_1. Assume that the attack set includes the first q targets. To add the next target, the attacker's payoff for all targets in Γ_i must be reduced to $U_i^a(c_{q+1}, t_{q+1})$ (Line 11). However, it might not be possible to do this. Once a target t is fully covered by the defender, there is no way to decrease the attacker's payoff below $U_i^{c,a}(t)$. Thus, if $\max_{1 \leq t \leq q} U_i^{c,a}(t) > U_i^a(c_{q+1}, t_{q+1})$ (Line 7), then it is impossible to induce the adversary i to attack target t_{q+1}. In that case, we must reduce the attacker's payoff for targets in the attack set to $\max_{1 \leq t \leq q} U_i^{c,a}(t)$ (Line 8). Then for each target $t \in \Gamma_i$, we compute the amount of additional coverage, $addCov[t]$, necessary to reach the required attacker payoff (Line 13). If the total amount of additional coverage exceeds the amount of remaining coverage, then $addedCov$ is recomputed and each target in the attack set is assigned ratio of the remaining coverage so to maintain the attack set (Line 17). There is then a check to see if $\mathbf{c} + addedCov$ satisfies b_i (Line 18). If b_i is still not satisfied, then the coverage $\mathbf{c}$ is updated to include $addedCov$ (Line 26) and the process is repeated for the next target (Line 28).

Algorithm 4: MIN-COV$(i, \mathbf{c}, \mathbf{b})$

```
1  Input: Game index $i$, initial coverage $\mathbf{c}$, lower bound $\mathbf{b}$;
2  $\mathbf{c}^* \leftarrow null$;
3  $minResources \leftarrow m$;
4  foreach $t' \in \Gamma_i(\mathbf{c})$ do
5      $\mathbf{c}' \leftarrow \mathbf{c}$ ;
6      $c'_{t'} = \frac{b_i - U_i^{u,a}(t')}{U_i^{c,a}(t') - U_i^{u,a}(t')}$;
7      foreach $t \in T \setminus \{t'\}$ do
8          if $U_i^a(c'_t, t) > U_i^a(c'_{t'}, t')$ then
9              $c'_t = \frac{U_i^a(c'_{t'}, t') - U_i^{u,a}(t)}{U_i^{c,a}(t) - U_i^{u,a}(t)}$;
10     if $U_i^d(\mathbf{c}') \geq b_i$ and $\sum_{t \in T} c'_t \leq minResources$ then
11         $\mathbf{c}^* \leftarrow \mathbf{c}'$;
12         $minResources \leftarrow \sum_{t \in T} c'_t$ ;
13 return $\mathbf{c}^*$
```

Then if $\mathbf{c} + addedCov$ expands Γ_i and exceeds b_i, it may be possible to use less defender resources and still satisfy b_i. Thus we use the algorithm MIN-COV to compute, $\forall t' \in \Gamma_i$, the amount of coverage needed to induce an attack on t' which yields a defender payoff of b_i. For each t', MIN-COV generates a defender coverage vector $\mathbf{c}'$, which is initialized to the current coverage $\mathbf{c}$. Coverage $c'_{t'}$ is updated such that the defender payoff for t' is b_i, yielding an attacker payoff $U_i^a(c'_{t'}, t')$ (Line 6). The coverage for every other target $t \in T \setminus \{t'\}$ is updated, if needed, to ensure that t' remains in Γ_i, i.e. $U_i^a(c'_{t'}, t') \geq U_i^a(c'_t, t)$ (Line 9). After this process, $\mathbf{c}'$ is guaranteed to satisfy b_i. From the set of defender coverage vectors, MIN-COV returns the $\mathbf{c}'$ which uses the least amount of defender resources. If while computing the additional coverage to added, either Γ_i is the set of all targets or all m security resources are exhausted, then both b_i and the CSOP are infeasible.

If $\mathbf{b}$ is satisfiable, ORIGAMI-M will return the minimum coverage vector $\mathbf{c}^*$ that satisfies $\mathbf{b}$. This coverage vector can be used to replace Equation (8) with $c^*_t \leq c_t \leq 1$.

7. ORIGAMI-A

In the previous section, we showed heuristics to improve the efficiency of our MILP approach. However, solving MILPs, even when constrained, is computationally expensive. Thus, we present ORIGAMI-A, an extension to ORIGAMI-M which eliminates the computational overhead of MILPs for solving CSOPs. The key idea of ORIGAMI-A is to translate a CSOP into a feasibility problem which can be solved using ORIGAMI-M. We then generate a series of these feasibility problems using binary search in order to approximate the optimal solution to the CSOP. As a result, this algorithmic approach is much more efficient.

ORIGAMI-M computes the minimum coverage vector necessary to satisfy a set of lower bound constraints $\mathbf{b}$. As our MILP approach is an optimization problem, lower bounds are specified for the secondary objectives but not the primary objective. We can convert this optimization problem into a feasibility problem by creating a new set of lower bounds constraints $\mathbf{b}^+$ by adding a lower bound constraint b_1^+ for the primary objective to the constraints $\mathbf{b}$. We set $b_1^+ = \min_{t \in T} U_1^{u,d}(t)$, the lowest defender payoff for leaving a target uncovered. Now instead of finding the coverage $\mathbf{c}$ which maximizes $U_1^d(\mathbf{c})$ and satisfies $\mathbf{b}$, we can use ORIGAMI-M to determine if there exists a coverage vector $\mathbf{c}$ such that $\mathbf{b}^+$ is satisfied.

Algorithm 5: ORIGAMI-A$(\mathbf{b}, \alpha)$

```
1  $\mathbf{c} \leftarrow$ empty coverage vector;
2  $b_1^+ \leftarrow \min_{t \in T} U_1^{u,d}(t)$;
3  $\mathbf{b}^+ \leftarrow \{b_1^+\} \cup \mathbf{b}$ ;
4  for $1 \leq i \leq n$ do
5      $lower \leftarrow b_i^+$;
6      $upper \leftarrow \max_{t \in T} U_i^{c,d}(t)$;
7      while $upper - lower > \alpha$ do
8          $b_i^+ \leftarrow \frac{upper + lower}{2}$;
9          $\mathbf{c}' \leftarrow$ ORIGAMI-M$(\mathbf{b}^+)$;
10         if $\mathbf{c}' = violated$ then
11             $upper \leftarrow b_i^+$;
12         else
13             $\mathbf{c} \leftarrow \mathbf{c}', lower \leftarrow b_i^+$;
14     $b_i^+ \leftarrow U_i^d(\mathbf{c})$;
15 return $\mathbf{c}$ ;
```

ORIGAMI-A finds an approximately optimal coverage vector $\mathbf{c}$ by using ORIGAMI-M to solve a series of feasibility problems. This series is generated by sequentially performing binary search on the objectives starting with initial lower bounds defined in $\mathbf{b}^+$. For objective i, the lower and upper bounds for the binary search are, respectively, b_i^+ and $\max_{t \in T} U_1^{c,d}(t)$, the highest defender payoff for covering a target. At each iteration, $\mathbf{b}^+$ is updated by setting $b_i^+ = (upper + lower)/2$ and then passed as input to ORIGAMI-M. If $\mathbf{b}^+$ is found to be feasible, then the lower bound is updated to b_i^+ and $\mathbf{c}$ is updated to the output of ORIGAMI-M, otherwise the upper bound is updated to b_i^+. This process is repeated until the difference between the upper and lower bounds reaches the termination threshold, α. Before proceeding to the next objective, b_i^+ is set to $U_i^d(\mathbf{c})$ in case the binary search terminated on an infeasible problem. After searching over each objective, ORIGAMI-A will return a coverage vector $\mathbf{c}$ such that $U_1^d(\mathbf{c}^*) - U_1^d(\mathbf{c}) \leq \alpha$, where $\mathbf{c}^*$ is the optimal coverage vector for a CSOP defined by $\mathbf{b}$.

The solutions found by ORIGAMI-A are no longer Pareto optimal. Let Ω_α be the objective space of the solutions found by ORIGAMI-A. We can bound its efficiency loss using the approximation measure $\rho(\epsilon, \alpha) = \max_{\mathbf{v} \in \Omega} \min_{\mathbf{v}' \in \Omega_\alpha} \max_{1 \leq i \leq n}(v_i - v'_i)$.

THEOREM 5. $\rho(\epsilon, \alpha) \leq \max\{\epsilon, \alpha\}$.

PROOF. Similar to the proof of Theorem 4, for each point $\mathbf{v} \in \Omega$, we can use Algorithm 2 to find a CSOP with constraints $\mathbf{b}$ which is solved using ORIGAMI-A with coverage $\mathbf{c}$ such that 1) $b_i \leq v_i$ for $i > 1$ and 2) $v'_i \geq v_i - \epsilon$ for $i > 1$ where $\mathbf{v}' = U^d(\mathbf{c})$.

Assume that the optimal coverage is $\mathbf{c}^*$ for the CSOP with constraints $\mathbf{b}$. It follows that $U_1^d(\mathbf{c}^*) \geq v_1$ since the coverage resulting in point $\mathbf{v}$ is a feasible solution to the CSOP with constraints $\mathbf{b}$. ORIGAMI-A will terminate if the difference between lower bound and upper bound is no more than α. Therefore, $v'_1 \geq U_1^d(\mathbf{c}^*) - \alpha$. Combining the two results, it follows that $v'_1 \geq v_1 - \alpha$.

Therefore, for any point missing in the frontier $\mathbf{v} \in \Omega$, we can find a point $\mathbf{v}' \in \Omega_\alpha$ such that 1) $v'_1 \geq v_1 - \alpha$ and $v'_i \geq v_i - \epsilon$ for $i > 1$. It then follows that $\rho(\epsilon, \alpha) \leq \max\{\epsilon, \alpha\}$. □

8. EVALUATION

We perform our evaluation by running the full algorithm in order to generate the Pareto frontier for randomly-generated MOSGs. For our experiments, the defender's covered payoff $U_i^{c,d}(t)$ and attacker's uncovered payoff $U_i^{u,a}(t)$ are uniformly distributed integers between 1 and 10 for all targets. Conversely, the defender's uncovered payoff $U_i^{u,d}(t)$ and attacker's covered payoff $U_i^{c,a}(t)$

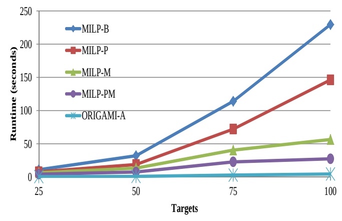


Figure 4: Scaling up targets

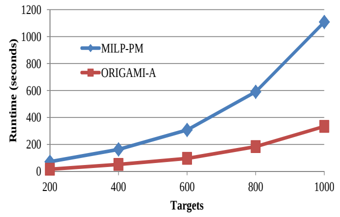


Figure 5: More target scale up

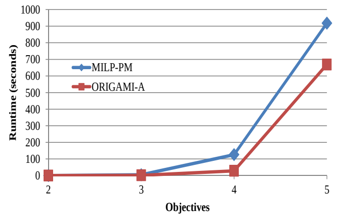


Figure 6: Scaling up objectives

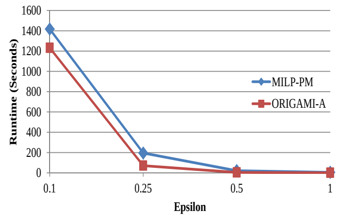


Figure 7: Scaling down epsilon

are uniformly distributed integers between -1 and -10. Unless otherwise mentioned, the setup for each experiment is 3 objectives, 25 targets, $\epsilon = 1.0$, and $\alpha = 0.001$. The amount of defender resources m is fixed at 20% of the number of targets. For experiments comparing multiple formulations, all formulations were tested on the same set of MOSGs. A maximum cap on runtime for each sample is set at 1800 seconds. We solved our MILP formulations using CPLEX version 12.1. The results were averaged over 30 trials.

8.1 Runtime Analysis

We evaluted five MOSG formulations. We refer to the baseline MILP formulation as MILP-B. The MILP formulation adding a bound on the defender's payoff for the primary objective is MILP-P. MILP-M uses ORIGAMI-M to compute bounds on defender coverage. MILP-P can be combined with MILP-M to form MILP-PM. The algorithmic approach using ORIGAMI-A will be referred to by name. For the number of targets, we evaluate all five formulations for solving CSOPs. We then select ORIGAMI-A and the fastest MILP formulation, MILP-PM, to evaluate the remaining factors.

Effect of the Number of Targets:This section presents results showing the efficiency of our different formulations as the number of targets is increased. In Figure 4, the x-axis represents the number of the targets in the MOSG. The y-axis is the number of seconds needed by Iterative ϵ-Constraints to generate the Pareto frontier using the different formulations for solving CSOPs. Our baseline MILP formulation, MILP-B, has the highest runtime for each number of targets we tested. By adding an upper bound on the defender payoff for the primary objective, MILP-P yields a runtime savings of 36% averaged over all numbers of targets compared to MILP-B. MILP-M uses ORIGAMI-M to compute lower bounds for defender coverage, resulting in a reduction of 70% compared to MILP-B. Combining the insights from MILP-P and MILP-M, MILP-PM achieves an even greater reduction of 82%. Removing the computational overhead of solving MILPs, ORIGAMI-A is the most efficient formulation with a 97% reduction. For 100 targets, ORIGAMI-A requires 4.53 seconds to generate the Pareto frontier, whereas the MILP-B takes 229.61 seconds, a speedup of >50 times. Even compared to fastest MILP formulation, MILP-PM at 27.36 seconds, ORIGAMI-A still achieves a 6 times speedup. T-test yields p-value<0.001 for all comparison of different formulations when there are 75 or 100 targets.

We conducted an additional set of experiments to determine how MILP-PM and ORIGAMI-A scale up for an order of magnitude increase in the number of targets by testing on MOSGs with between 200 and 1000 targets. Based on the trends seen in the data, we can concluded that ORIGAMI-A significantly outperforms MILP-PM for MOSGs with large number of targets. Therefore, the number of targets in an MOSG is not a prohibitive bottleneck for generating the Pareto frontier using ORIGAMI-A.

Effect of the Number of Objectives: Another key factor on the efficiency of Iterative ϵ-Constraints is the number of objectives which determines the dimensionality of the objective space that Iterative ϵ-Constraints must search. We ran experiments for MOSGs with between 2 and 6 objectives. For these experiments, we fixed the number of targets at 10. Figure 6 shows the effect of scaling up the number of objectives. The x-axis represents the number of objectives, whereas the y-axis indicates the average time needed to generate the Pareto frontier. For both MILP-PM and ORIGAMI-A, we observe an exponential increase in runtime as the number of objectives is scaled up. For both approaches, the Pareto frontier can be computed in under 5 seconds for 2 and 3 objectives. Whereas, with 6 objectives neither approach is able to generate the Pareto frontier before the runtime cap of 1800 seconds. These results show that the number of objectives, and not the number of targets, is the key limiting factor in solving MOSGs.

Effect of Epsilon: A third critical factor on the running time of Iterative ϵ-Constraints is the value of the ϵ parameter which determines the granularity of the search process through the objective space. In Figure 7, results are shown for ϵ values of 0.1, .25, .5, and 1.0. Both MILP-PM and ORIGAMI-A see a sharp increase in runtime as the value of ϵ is decreased due to the rise in the number of CSOPs solved. For example, with $\epsilon = 1.0$ the average Pareto frontier consisted of 49 points, whereas for $\epsilon = 0.1$ that number increased to 8437. Due to the fact that ϵ is applied to the $n-1$ dimensional objective space, the increase in the runtime resulting from decreasing ϵ is exponential in the number of secondary objectives. Thus, using small values of ϵ can be computationally expensive, especially if the number of objectives is large.

Effect of the Similarity of Objectives: In previous experiments, all payoffs were sampled from a uniform distribution resulting in independent objective functions. However, it is possible that in a security setting, the defender could face multiple attacker types which share certain similarities, such as the same relative preferences over a subset of targets. To evaluate the effect of objective similarity on runtime, we used a single security game to create a Gaussian function with standard deviation σ from which all the payoffs for an MOSG are sampled. Figure 8 shows the results for using ORIGAMI-A to solve MOSGs with between 3 and 7 objectives using σ values between 0 and 2.0 as well as for uniformly distributed objectives. For $\sigma = 0$, the payoffs for all security games are the same, resulting in Pareto frontier consisting of a single point. In this extreme example, the number of objectives does not impact the runtime. However, as the number of objectives increases, less dissimilarity between the objectives is needed before the runtime starts increasing dramatically. For 3 and 4 objectives, the amount of similarity has negligible impact on runtime. The experiments with 5 objectives time out after 1800 seconds for the uniformly distributed objectives. Whereas, 6 objectives times out at $\sigma = 1.0$ and 7 objectives at $\sigma = 0.5$. We conclude that it is possible to scale to larger number of objectives if there is similarity between the attacker types.

8.2 Solution Quality Analysis

Effect of Epsilon: If the Pareto frontier is continuous, only a subset of that frontier can be generated. Thus, it is possible that one of the Pareto optimal points not generated by Iterative ϵ-Constraints

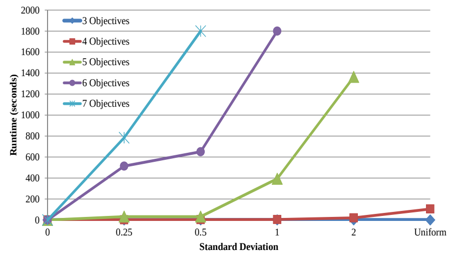


Figure 8: Objective similarity

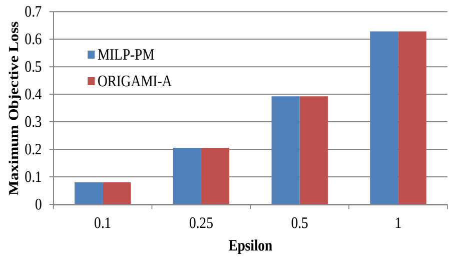


Figure 9: Epsilon solution quality

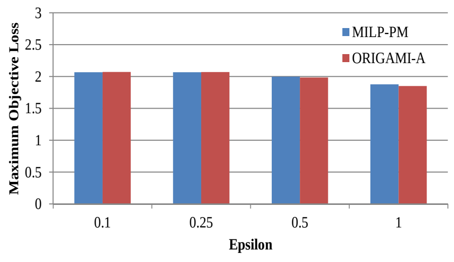


Figure 10: Comparison against uniformly weighted Bayesian security games

would be the most preferred solution, were it presented to the end user. In Section 5.2, we proved that the maximum utility loss for each objective resulting from this situation could be bounded by ϵ. We conducted experiments to empirically verify our bounds and to determine if the actual maximum objective loss was less than ϵ.

Ideally, we would compare the Pareto frontier generated by Iterative ϵ-Constraints to the true Pareto frontier. However, the true Pareto frontier may be continuous and impossible for us to generate, thus we simulate the true frontier by using $\epsilon = 0.001$. Due to the computational complexity associated with such a value of ϵ, we fix the number of objectives to 2. Figure 9 shows the results for ϵ values of 0.1, .25, .5, and 1.0. The x-axis represent the value of ϵ, whereas the y-axis represents the maximum objective loss when comparing the generated Pareto frontier to the true Pareto frontier. We observe that the maximum objective loss is less than ϵ for each value of ϵ tested. At $\epsilon = 1.0$, the average maximum objective loss is only 0.63 for both MILP-PM and ORIGAMI-A. These results verify that the bounds for our algorithms are correct and that in practice we are able to generate a better approximation of the Pareto frontier than the bounds would suggest.

Comparison against Uniform Weighting: We introduced the MOSG model, in part, because it eliminates the need to specify a probability distribution over attacker types a priori. However, even if the probability distribution is unknown it is still possible to use the Bayesian security game model with a uniform distribution. We conducted experiments to show the potential benefit of using MOSG over Bayesian security games in such cases. We computed the maximum objective loss sustained by using the Bayesian solution as opposed to a point in the Pareto frontier generated by Iterative ϵ-Constraints. If $\mathbf{v}'$ is the solution to a uniformly weighted Bayesian security game then the equation for maximum objective loss is $\max_{v \in \Omega_\epsilon} \max_i (v_i - v'_i)$. Figure 10 shows the results for ϵ values of 0.1, .25, .5, and 1.0. At $\epsilon = 1.0$, the maximum objective loss were 1.87 and 1.85 for MILP-PM and ORIGAMI-A. Decreasing ϵ all the way to 0.1 increases the maximum objective loss by less than 12% for both algorithms. These results suggests that ϵ has limited impact on maximum objective loss, which is a positive result as it implies that solving an MOSG with a large ϵ can still yield benefits over a uniform weighted Bayesian security game.

9. CONCLUSION

We built upon insights from game theory and multi-objective optimization to introduce a new model, multi-objective security games (MOSG), for domains where security forces must balance multiple objectives. Contributions include: 1) Iterative ϵ-Constraints, a high-level approach for transforming MOSGs into a sequence of CSOPs, 2) exact MILP formulations, both with and without heuristics, for solving CSOPs, and 3) ORIGAMI-A, an approximate approach for solving CSOPs. We then provided bounds for both the complexity as well as the solution quality of our approaches; additionally we provided detailed experimental comparison of the different approaches presented.

10. ACKNOWLEDGEMENT

This research was supported by the United States Department of Homeland Security through the National Center for Border Security and Immigration (NCBSI).

11. REFERENCES

[1] B. An, J. Pita, E. Shieh, M. Tambe, C. Kiekintveld, and J. Marecki. Guards and protect: next generation applications of security games. *ACM SIGecom Exchanges*, 10(1):31–34, 2011.

[2] N. Basilico, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS*, pages 57–64, 2009.

[3] K. Bringmann, T. Friedrich, F. Neumann, and M. Wagner. Approximation-guided evolutionary multi-objective optimization. In *IJCAI*, pages 1198–1203, 2011.

[4] V. Chankong and Y. Haimes. *Multiobjective decision making: theory and methodology*, volume 8. North-Holland New York, 1983.

[5] V. Conitzer and D. Korzhyk. Commitment to correlated strategies. In *AAAI*, pages 632–637, 2011.

[6] K. Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. Wiley, 2001.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Trans. on Evolutionary Computation*, 6(2):182–197, 2002.

[8] M. Jain, J. Tsai, J. Pita, C. Kiekintveld, S. Rathi, M. Tambe, and F. Ordonez. Software Assistants for Randomized Patrol Planning for the LAX Airport Police and the Federal Air Marshals Service. *Interfaces*, 40:267–290, 2010.

[9] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, F. Ordonez, and M. Tambe. Computing optimal randomized resource allocations for massive security games. In *AAMAS*, pages 689–696, 2009.

[10] D. Korzhyk, V. Conitzer, and R. Parr. Security games with multiple attacker resources. In *IJCAI*, pages 273–279, 2011.

[11] M. Laumanns, L. Thiele, and E. Zitzler. An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, 169(3):932–942, 2006.

[12] G. Mavrotas. Effective implementation of the [epsilon]-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, 213(2):455–465, 2009.

[13] P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordonez, and S. Kraus. Playing games with security: An efficient exact algorithm for bayesian stackelberg games. In *AAMAS*, pages 895–902, 2008.

[14] B. von Stengel and S. Zamir. Leadership with commitment to mixed strategies. Technical Report LSE-CDAM-2004-01, CDAM Research Report, 2004.

Strategy Purification and Thresholding: Effective Non-Equilibrium Approaches for Playing Large Games[*]

Sam Ganzfried, Tuomas Sandholm, and Kevin Waugh
Computer Science Department
Carnegie Mellon University
{sganzfri, sandholm, waugh}@cs.cmu.edu

ABSTRACT

There has been significant recent interest in computing effective strategies for playing large imperfect-information games. Much prior work involves computing an approximate equilibrium strategy in a smaller abstract game, then playing this strategy in the full game (with the hope that it also well approximates an equilibrium in the full game). In this paper, we present a family of modifications to this approach that work by constructing non-equilibrium strategies in the abstract game, which are then played in the full game. Our new procedures, called *purification* and *thresholding*, modify the action probabilities of an abstract equilibrium by preferring the higher-probability actions. Using a variety of domains, we show that these approaches lead to significantly stronger play than the standard equilibrium approach. As one example, our program that uses purification came in first place in the two-player no-limit Texas Hold'em total bankroll division of the 2010 Annual Computer Poker Competition. Surprisingly, we also show that purification significantly improves performance (against the full equilibrium strategy) in random 4×4 matrix games using random 3×3 abstractions. We present several additional results (both theoretical and empirical). Overall, one can view these approaches as ways of achieving robustness against overfitting one's strategy to one's lossy abstraction. Perhaps surprisingly, the performance gains do not necessarily come at the expense of worst-case exploitability.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; J.4 [**Social and Behavioral Sciences**]: Economics

[*]This material is based upon work supported by the National Science Foundation under grants IIS-0964579, IIS-0905390, and CCF-1101668. We also acknowledge Intel Corporation and IBM for their machine gifts. Early versions of this paper appeared as "Strategy Purification" in the National Conference on Artificial Intelligence (AAAI) Workshop on Applied Adversarial Reasoning and Risk Modeling, 2011, and as a 2-page abstract in the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2011.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

General Terms

Algorithms, Economics, Theory

Keywords

Game theory, game playing, large games, abstraction, reverse mapping, non-equilibrium approaches, purification

1. INTRODUCTION

Developing effective strategies for agents in multiagent systems is an important and challenging problem. It has received significant attention in recent years from several different communities—in part due to the competitions held at top conferences (e.g, the computer poker, robo-soccer, and trading agent competitions). As many domains are so large that solving them directly (i.e., computing a Nash equilibrium or a solution according to some other relevant solution concept) is computationally infeasible, some amount of approximation is necessary to produce agents.

Specifically, significant work has been done on computing approximate game-theory-based strategies in large imperfect-information games. This work typically follows a three-step approach, which is depicted in Figure 1. First, an *abstraction algorithm* is run on the original game G to construct a smaller game G' which is strategically similar to G [1, 2, 3, 11]. Second, an *equilibrium-finding algorithm* is run on G' to compute an ϵ-equilibrium σ' [6, 13]. Third, a *reverse mapping* is applied to σ' to compute an approximate equilibrium σ in the full game G [5, 10]. While most prior work has focused on the first two steps of this approach, in this paper we focus on the third. In particular, we propose first mapping the abstract approximate-equilibrium strategy profile to a *non-equilibrium* strategy profile in the abstract game, which we then map to a strategy profile in the full game.

Almost all prior work has used the trivial reverse mapping in which σ is the straightforward projection of σ' into G. In other words, once the abstract game is solved, its solution is just played directly in the full game. In this paper, we show that applying more sophisticated reverse mappings can lead to significant performance improvements—even if they produce strategy profiles that are no longer equilibria in the abstract game.

One of the key ideas that motivated our approach is that the exact action probabilities of a mixed strategy equilibrium in an abstraction can exemplify overfitting to the particular abstraction used. (Our results confirm this.) Ideally, we would like to extrapolate general principles from the strategy rather than just use values that were finely

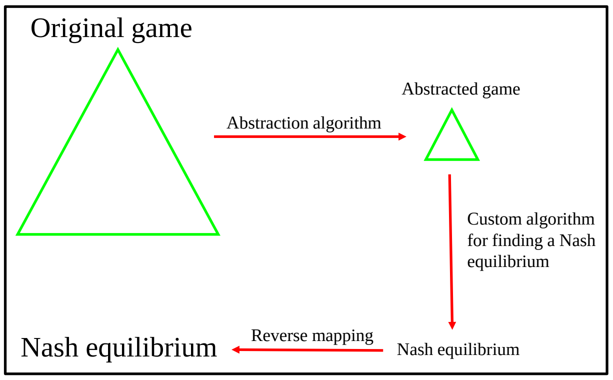


Figure 1: General approach for solving large games.

tuned for a specific abstraction. This is akin to the classic example from machine learning, where we would prefer a degree-one polynomial that fits the training data quite well to a degree-hundred polynomial that may fit it slightly better. (Subsequent to the appearance of the earlier versions of our paper, others have also shown that overfitting strategies to a particular abstraction is a very significant and real problem in large imperfect-information games [7].)

We present a family of modifications to the standard approach that work by constructing non-equilibrium strategies in the abstract game, which are then played in the full game. Our new procedures, called *purification* and *thresholding*, modify the action probabilities of an abstract equilibrium by placing a preference on the higher-probability actions. The main intuition behind our algorithms is that we should ignore actions that are played with small probability in the abstract equilibrium, as they are likely due to abstraction coarseness, overfitting, or failure of the equilibrium-finding algorithm to fully converge.

Using a variety of experimental domains, we show that our new approach leads to significantly stronger play than the standard abstraction/equilibrium approach. For example, our program that uses purification won the two-player no-limit Texas Hold'em total bankroll division of the 2010 Annual Computer Poker Competition (ACPC), held at AAAI. Surprisingly, we also show that purification significantly improves performance (against the full equilibrium strategy) in random 4×4 matrix games using random 3×3 abstractions. We present additional results (both theoretical and empirical), including: worst-case theoretical results, empirical and theoretical results on specific support properties for which purification helps in matrix games, and experimental results in well-studied large imperfect-information games (Leduc Hold'em and Texas Hold'em).

2. GAME THEORY BACKGROUND

In this section, we briefly review relevant definitions and prior results from game theory and game solving.

2.1 Strategic-form games

The most basic game representation, and the standard representation for simultaneous-move games, is the *strategic form*. A *strategic-form game* (aka matrix game) consists of a finite set of players N, a space of *pure strategies* S_i for each player, and a utility function $u_i : \times S_i \to \mathbb{R}$ for each player. Here $\times S_i$ denotes the space of *strategy profiles*—vectors of pure strategies, one for each player.

The set of *mixed strategies* of player i is the space of probability distributions over his pure strategy space S_i. We will denote this space by Σ_i. Define the *support* of a mixed strategy to be the set of pure strategies played with nonzero probability. If the sum of the payoffs of all players equals zero at every strategy profile, then the game is called *zero sum*. In this paper, we will be primarily concerned with two-player zero-sum games; we will show that the new approaches lead to performance improvements even in this class of games where the equilibrium approach should be at its best. If the players are following strategy profile σ, we let σ_{-i} denote the strategy taken by player i's opponent, and we let Σ_{-i} denote the opponent's entire mixed strategy space.

2.2 Extensive-form games

An *extensive-form* game is a general model of multiagent decision making with potentially sequential and simultaneous actions and imperfect information. As with perfect-information games, extensive-form games consist primarily of a game tree; each non-terminal node has an associated player (possibly *chance*) that makes the decision at that node, and each terminal node has associated utilities for the players. Additionally, game states are partitioned into *information sets*, where the player whose turn it is to move cannot distinguish among the states in the same information set. Therefore, in any given information set, a player must choose actions with the same distribution at each state contained in the information set. If no player forgets information that he previously knew, we say that the game has *perfect recall*. A (behavioral) *strategy* for player i, $\sigma_i \in \Sigma_i$, is a function that assigns a probability distribution over all actions at each information set belonging to i.

2.3 Nash equilibria

Player i's *best response* to σ_{-i} is any strategy in

$$\arg \max_{\sigma'_i \in \Sigma_i} u_i(\sigma'_i, \sigma_{-i}).$$

A *Nash equilibrium* is a strategy profile σ such that σ_i is a best response to σ_{-i} for all i. An ϵ-*equilibrium* is a strategy profile in which each player achieves a payoff of within ϵ of his best response.

In two-player zero-sum games, we have the following result which is known as the *minimax theorem*:

$$v^* = \max_{\sigma_1 \in \Sigma_1} \min_{\sigma_2 \in \Sigma_2} u_1(\sigma_1, \sigma_2) = \min_{\sigma_2 \in \Sigma_2} \max_{\sigma_1 \in \Sigma_1} u_1(\sigma_1, \sigma_2).$$

We refer to v^* as the *value* of the game to player 1. Sometimes we will write v_i as the value of the game to player i. It is worth noting that in two-player zero-sum games, any equilibrium strategy for a player will guarantee an expected payoff of at least the value of the game to that player.

All finite games have at least one Nash equilibrium. In two-player zero-sum strategic-form games, a Nash equilibrium can be found efficiently by linear programming. In the case of zero-sum extensive-form games with perfect recall, there are efficient techniques for finding an ϵ-equilibrium, such as linear programming [8], generalizations of the excessive gap technique [6], and counterfactual regret minimization [13]. The latter two algorithms scale to games with approximately 10^{12} game tree states, while the best current general-purpose linear programming technique (CPLEX's barrier method) scales to games with around 10^8 states.

2.4 Abstraction

Despite the tremendous progress in equilibrium-finding in recent years, many interesting real-world games are so large that even the best algorithms have no hope of computing an equilibrium directly. The standard approach of dealing with this is to apply an *abstraction* algorithm, which constructs a smaller game that is similar to the original game; then the smaller game is solved, and its solution is mapped to a strategy profile in the original game. The approach has been applied to two-player Texas Hold'em poker, first with a manually generated abstraction [1], and now with abstraction algorithms [2]. Many abstraction algorithms work by coarsening the moves of chance, collapsing several information sets of the original game into single information sets of the abstracted game.

The game tree of two-player no-limit Texas Hold'em has about 10^{71} states (while that of two-player limit Texas Hold' em has about 10^{18} states); so significant abstraction is necessary.

3. PURIFICATION AND THRESHOLDING

Suppose we are playing a game Λ that is too large to solve directly. As described in Section 2.4, the standard approach would be to construct an abstract game Λ', compute an equilibrium σ' of Λ', then play the strategy profile σ induced by σ' in the full game Λ.

One possible problem with this approach is that the specific strategy profile σ' might be very finely tuned for the abstract game Λ', and it could perform arbitrarily poorly in the full game (see the results in Section 5). Ideally we would like to extrapolate the important features from σ' that will generalize to the full game and avoid playing a strategy that is overfit to the particular abstraction. This is one of the key motivations for our new approaches, *purification* and *thresholding.*

3.1 Purification

Let σ_i be a mixed strategy for player i in a strategic-form game, and let $S = \arg\max_j \sigma_i(j)$, where j ranges over all of player i's pure strategies. Then we define the *purification* $\text{pur}(\sigma_i)$ of σ_i as follows:

$$\text{pur}(\sigma_i)(j) = \left\{ \begin{array}{lcl} 0 & : & j \notin S \\ \frac{1}{|S|} & : & j \in S \end{array} \right.$$

If σ_i plays a single pure strategy with highest probability, then the purification will play that strategy with probability 1. If there is a tie between several pure strategies of the maximum probability played under σ_i, then the purification will randomize equally between all maximal such strategies. Thus the purification will usually be a pure strategy, and will only be a mixed strategy in degenerate special cases when several pure strategies are played with identical probabilities.

If σ_i is a behavioral strategy in an extensive-form game, we define the purification similarly; at each information set I, $\text{pur}(\sigma_i)$ will play the purification of σ_i at I.

3.2 Thresholding

The effects of purification can be quite extreme in some situations. For example, if σ_i plays action a with probability 0.51 and action b with probability 0.49, then b will never be played after purification. We also consider a more relaxed approach, called *thresholding*, that only eliminates actions below a prescribed ϵ to help alleviate this concern.

Thresholding works by setting all actions that have weight below ϵ to 0, then renormalizing the action probabilities. Pseudocode is given below in Algorithm 1. One intuitive interpretation of thresholding is that actions with probability below ϵ may just have been given positive probability due to noise from the abstraction (or because an equilibrium-finding algorithm had not yet taken those probabilities all the way to zero), and really should not be played in the full game. Additionally, low probability actions are often played primarily to protect a player from being exploited, and this may be an overstated concern against realistic opponents (as discussed further in Section 4.2).

Algorithm 1 Threshold(σ_i, ϵ)

```
for $j = 1$ to $|S|$ do
  if $\sigma_i(j) < \epsilon$ then
    $\sigma_i(j) \leftarrow 0$
  end if
end for
normalize($\sigma_i$)
return $\sigma_i$
```

4. EVALUATION METRICS

In recent years, several different metrics have been used to evaluate strategies in large games.

4.1 Empirical performance

The first metric, which is perhaps the most meaningful, is *empirical performance* against other realistic strategies. For example, in the ACPC, programs submitted from researchers and hobbyists from all over the world compete against one another. Empirical performance is the metric we will be using in Section 8 when we assess our performance in Texas Hold'em.

4.2 Worst-case exploitability

The worst-case *exploitability* of player i's strategy σ_i is the difference between the value of the game to player i and the payoff when the opponent plays his best response to σ_i (aka his *nemesis strategy*). Formally it is defined as follows:

$$\text{expl}(\sigma_i) = v_i - \min_{\sigma_{-i} \in \Sigma_{-i}} u_i(\sigma_i, \sigma_{-i}).$$

Worst-case exploitability has recently been used to assess strategies in several variants of poker [4, 7, 12].

Any equilibrium has zero exploitability, since it receives payoff v_i against its nemesis. So if our goal were to approximate an equilibrium of the full game, worst-case exploitability would be a good metric to use, since it approaches zero as the strategy approaches equilibrium.

Unfortunately, the worst-case exploitability metric has several drawbacks. First, it cannot be computed in very large games (though very recent advancements have made it possible to compute full best responses offline in two-player limit Texas Hold'em, which has about 10^{18} game states [7], and we will be leveraging that algorithm in our experiments).

Second, exploitability is a worst-case metric that assumes the opponent is able to *optimally* exploit us in the full game (i.e., he knows our full strategy and is able to efficiently compute a full best response in real time). In fact, it is quite common in very large games for agents to simply play static, fixed strategies the entire time, since the number of interactions is generally tiny compared to the size of the game, and it is usually quite difficult to learn to effectively exploit opponents online. For example, in recent computer poker competitions, almost all submitted programs simply play a fixed strategy. In the 2010 ACPC, many of the entrants attached summaries describing their algorithm. Of the 17 bots for which summaries were included, 15 played fixed strategies, while only 2 included some element of attempted exploitation. If the opponents are just playing a fixed strategy and not trying to exploit us, then worst-case exploitability is too pessimistic of an evaluation metric. Furthermore, if the opponents have computational limitations and use abstractions, then they will not be able to fully exploit us in the full game.

4.3 Performance against full equilibrium

In this paper, we will also evaluate strategies based on performance against equilibrium in the full game. The intuition behind this metric is that in many large two-player zero-sum games, the opponents are simply playing fixed strategies that attempt to approximate an equilibrium of the full game (using some abstraction). For example, most entrants in the ACPC do this. Against such static opponents, worst-case exploitability is not very significant, as the agents are not generally adapting to exploit us.

This metric, like worst-case exploitability, is not feasible to apply on very large games. However, we can still apply it to smaller games as a means of comparing different solution techniques. In particular, we will use this metric in Sections 6 and 7 when presenting our experimental results on random matrix games and Leduc Hold'em. This metric has similarly been used on solvable problem sizes in the past to compare abstraction algorithms [4].

5. THEORY: SELECTIVE SUPERIORITY

So which approach is best: purification, thresholding, or the standard abstraction/equilibrium approach? It turns out that using the performance against full equilibrium metric, there exist games for which each technique can outperform each other. Thus, from a worst-case perspective, not much can be said in terms of comparing the approaches.

Proposition 1 shows that, for *any* equilibrium-finding algorithm, there exists a game and an abstraction such that purification does arbitrarily better than the standard approach.

PROPOSITION 1. *For any equilibrium-finding algorithms A and A', and for any $k > 0$, there exists a game Λ and an abstraction Λ' of Λ, such that*

$$u_1(pur(\sigma_1'), \sigma_2) \geq u_1(\sigma_1', \sigma_2) + k,$$

where σ' is the equilibrium of Λ' computed by algorithm A', and σ is the equilibrium of Λ computed by A.

PROOF. Consider the game in Figure 2. Let Λ denote the full game, and let Λ' denote the abstraction in which player 2 (the column player) is restricted to only playing L or M, but the row player's strategy space remains the same. Then Λ' has a unique equilibrium in which player 1 plays U with probability $\frac{1}{3}$, and player 2 plays L with probability $\frac{1}{3}$. Since this is the unique equilibrium, it must be the one output by algorithm A'. Note that player 1's purification $\text{pur}(\sigma_1')$ of σ' is the pure strategy D.

	L	M	R
U	2	0	$-3k-1$
D	0	1	-1

Figure 2: Two-player zero-sum game used in the proof of Proposition 1.

Note that in the full game Λ, the unique equilibrium is (D,R), which we denote by σ. As before, since this equilibrium is unique it must be the one output by algorithm A. Then we have

$$\begin{aligned} u_1(\sigma_1', \sigma_2) = \frac{1}{3}(-3k-1) + \frac{2}{3}(-1) &= -k-1 \\ u_1(\text{pur}(\sigma_1'), \sigma_2) &= -1. \end{aligned}$$

So $u_1(\sigma_1', \sigma_2) + k = -1$, and therefore

$$u_1(\text{pur}(\sigma_1'), \sigma_2) = u_1(\sigma_1', \sigma_2) + k.$$

□

We can similarly show that purification can also do arbitrarily worse against the full equilibrium than standard unpurified abstraction, and that both procedures can do arbitrarily better or worse than thresholding (using any threshold cutoff). We can also show similar results using an arbitrary multiplicative (rather than additive) constant k.

6. RANDOM MATRIX GAMES

The first set of experiments we conduct to demonstrate the power of purification is on random matrix games. This is perhaps the most fundamental and easy to analyze class of games, and is a natural starting point when analyzing new algorithms.

6.1 Evaluation methodology

We study random 4×4 two-player zero-sum matrix games with payoffs drawn uniformly at random from [-1,1]. We repeatedly generated random games and analyzed them using the following procedure. First, we computed an equilibrium of the full 4×4 game Λ; denote this strategy profile by σ^F. Next, we constructed an abstraction Λ' of Λ by ignoring the final row and column of Λ. In essence, Λ' is a naïve, random abstraction of Λ, since there is nothing special about the final row or column. As in Λ, we computed an equilibrium σ^A of Λ'. We then compared $u_1(\sigma_1^A, \sigma_2^F)$ to $u_1(\text{pur}(\sigma_1^A), \sigma_2^F)$ to determine the effect of purification on performance of the abstract equilibrium strategy for player 1 against the full equilibrium strategy of player 2.

To solve the full and abstract games, we used two different procedures. For our first set of experiments comparing the overall performance of purified vs. unpurified abstract equilibrium strategies, we used a standard algorithm involving solving a single linear program [8]. For our results on supports, we used a custom support enumeration algorithm (similar to the approach of Porter et al. [9]). We note that it is possible that the specific algorithm used may have a significant effect on the results (i.e., certain algorithms may be more likely to select equilibria with specific properties when several equilibria exist).

6.2 Experimental results and theory

In our experiments on 4×4 random games, we performed 1.5 million trials; the results are given in Table 1. The first row gives the average value of $u_1(\text{pur}(\sigma_1^A), \sigma_2^F)$ over all trials, while the second row gives the average value of $u_1(\sigma_1^A, \sigma_2^F)$. We conclude that purified abstraction outperforms the standard unpurified abstraction approach using 95% confidence intervals.

The next three rows of Table 1 report the number of trials for which purification led to an increased, decreased, or unchanged payoff of the abstract equilibrium strategy of player 1 against the full equilibrium strategy of player 2. While purification clearly improved performance more often than it hurt performance (17.44% vs. 11.48%), for the overwhelming majority of cases it led to no change in performance (71.08%). In particular, Proposition 2 gives two general sets of conditions under which purification leads to no change in performance.

Proposition 2. *Let Λ be a two-player zero-sum game, and let Λ' be an abstraction of Λ. Let σ^F and σ^A be equilibria of Λ and Λ' respectively. Then*

$$u_1(\sigma_1^A, \sigma_2^F) = u_1(pur(\sigma_1^A), \sigma_2^F)$$

if either of the following conditions is met:

1. *σ^A is a pure strategy profile*
2. *$support(\sigma_1^A) \subseteq support(\sigma_1^F)$*

Proof. If the first condition is met, then $\text{pur}(\sigma_1^A) = \sigma_1^A$ and we are done. Now suppose the second condition is true and let $s, t \in support(\sigma_1^A)$ be arbitrary. This implies that $s, t \in support(\sigma_1^F)$ as well, which means that $u_1(s, \sigma_2^F) = u_1(t, \sigma_2^F)$, since a player is indifferent between all pure strategies in his support at an equilibrium. Since s and t were arbitrary, player 1 is also indifferent between all strategies in $support(\sigma_1^A)$ when player 2 plays σ_2^F. Since purification will just select one strategy in $support(\sigma_1^A)$, we are done. $\square$

To understand our results further, we investigated whether they would vary for different supports of σ^F. In particular, we kept separate tallies of the performance of $\text{pur}(\sigma_1^A)$ and σ_1^A for each support of σ^F. We observed that $\text{pur}(\sigma_1^A)$ outperformed σ_1^A on many of the supports, while they performed equally on some (and σ_1^A did not outperform $\text{pur}(\sigma_1^A)$ on any). These results are all statistically significant using 95% confidence intervals. A summary of the results from these experiments is given in Observation 1.

Observation 1. *In random 4×4 matrix games using 3×3 abstractions, $pur(\sigma_1^A)$ performs better than σ_1^A using a 95% confidence interval for each support of σ^F except for supports satisfing one of the following conditions, in which case neither $pur(\sigma_1^A)$ nor σ_1^A performs significantly better:*

1. *σ^F is the pure strategy profile in which each player plays his fourth pure strategy*
2. *σ^F is a mixed strategy profile in which player 1's support contains his fourth pure strategy, and player 2's support does not contain his fourth pure strategy.*

To interpret Observation 1, consider the following example. Suppose the support for player 1 includes his first three pure strategies, while the support for player 2 includes his final three pure strategies; denote this support profile by S^*. Now consider the set U of all games for which our equilibrium-finding algorithm outputs an equilibrium profile σ^F with support profile S^*. Since S^* does not satisfy either condition of Observation 1, this means that, in expectation over the set of all games in U,

$$u_1(\text{pur}(\sigma_1^A), \sigma_2^F) > u_1(\sigma_1^A, \sigma_2^F)$$

(i.e., purification improves the performance of the abstracted equilibrium strategy of player 1 against the full equilibrium strategy of player 2).

We find it interesting that there is such a clear pattern in the support structures for which $\text{pur}(\sigma_1^A)$ outperforms σ_1^A. We obtained identical results using 3×3 games with 2×2 abstractions. We did not experiment on games larger than 4×4. While we presented experimental results that are statistically significant at the 95% confidence interval, rigorously proving that the results of Observation 1 hold even on 4×4 games with 3×3 abstractions remains a challenging open problem. Resolving this problem would shed some light on the underlying reasons behind the observed performance improvements of purification in random matrix games, which are quite surprising and unintuitive. In addition, we conjecture that a more general theoretical result will hold for general matrix games with any size, using any size random abstractions. Proving such a result could have significant theoretical and practical implications.

7. LEDUC HOLD'EM

Leduc Hold'em is a simplified poker variant that has been used in previous work to evaluate imperfect-information game-playing techniques (e.g., [12]). Leduc Hold'em is large enough that abstraction has a non-trivial impact, but unlike larger games of interest (e.g., Texas Hold'em) it is small enough that equilibrium solutions in the full game can be quickly computed. That is, Leduc Hold'em allows for rapid and thorough evaluation of game-playing techniques against a variety of opponents, including an equilibrium opponent or a best responder.

Prior to play, a deck of six cards containing two Jacks, two Queens, and two Kings is shuffled and each player is dealt a single private card. After a round of betting, a public card is dealt face up for both players to see. If either player pairs this card, he wins at showdown; otherwise the player with the higher ranked card wins. For a complete description of the betting, we refer the reader to Waugh et al. [12].

7.1 Experimental evaluation and setup

To evaluate the effects of purification and thresholding in Leduc Hold'em, we compared the performance of a number of abstract equilibrium strategies altered to varying degrees by thresholding against a single equilibrium opponent averaged over both positions. The performance of a strategy (denoted EV for expected value) was measured in millibets per hand (mb/h), where one thousand millibets is a small

$u_1(\text{pur}(\sigma_1^A), \sigma_2^F)$ (purified average payoff)	-0.050987 ± 0.00042
$u_1(\sigma_1^A, \sigma_2^F)$ (unpurified average payoff)	-0.054905 ± 0.00044
Number of games where purification led to improved performance	261569 (17.44%)
Number of games where purification led to worse performance	172164 (11.48%)
Number of games where purification led to no change in performance	1066267 (71.08%)

Table 1: Results for experiments on 1.5 million random 4×4 matrix games using random 3×3 abstractions. The $\pm$ given is the 95% confidence interval.

bet. As the equilibrium opponent is optimal, the best obtainable performance is 0 mb/h. Note that the expected value computations in this section are exact.

We used card abstractions mimicking those produced by state-of-the-art abstraction techniques to create our abstract equilibrium strategies. Specifically, we used the five Leduc Hold'em card abstractions from prior work [12], denoted *JQK*, *JQ.K*, *J.QK*, *J.Q.K* and *full*. The abstraction *full* denotes the null abstraction (i.e., the full unabstracted game). The names of the remaining abstractions consist of groups of cards separated by periods. All cards within a group are indistinguishable to the player prior to the flop. For example, when a player using the *JQ.K* abstraction is dealt a card, he will know only if that card is a king, or if it is not a king. These abstractions can only distinguish pairs on the flop. By pairing these five card abstractions, one abstraction per player, we learned twenty four abstract equilibrium strategies using linear programming techniques. For example, the strategy *J.Q.K-JQ.K* denotes the strategy where our player of interest uses the *J.Q.K* abstraction and he assumes his opponent uses the *JQ.K* abstraction.

7.2 Purification vs. no purification

In Table 2 we present the performance of the regular and purified abstract equilibrium strategies against the equilibrium opponent. We notice that purification improves the performance in all but 5 cases. In many cases this improvement is quite substantial. In the cases where it does not help, we notice that at least one of the players is using the *JQK* card abstraction, the worst abstraction in our selection. Prior to purification, the best abstract equilibrium strategy loses at 43.8 mb/h to the equilibrium opponent. After purification, 14 of the 24 strategies perform better than the best unpurified strategy, the best of which loses at only 1.86 mb/h. That is, only five of the strategies that were improved by purification failed to surpass the best unpurified strategy.

7.3 Purification vs. thresholding

In Figure 3 we present the results of three abstract equilibrium strategies thresholded to varying degrees against the equilibrium opponent. We notice that, the higher the threshold used the better the performance tends to be. Though this trend is not monotonic, all the strategies that were improved by purification obtained their maximum performance when completely purified. Most strategies tended to improve gradually as the threshold was increased, but this was not the case for all strategies. As seen in the figure, the *JQ.K-JQ.K* strategy spikes in performance between the thresholds of 0.1 and 0.15.

From these experiments, we conclude that purification tends to improve the performance of an abstract equilibrium strategy against an unadaptive equilibrium opponent in Leduc Hold'em. Though thresholding is itself helpful, it

Strategy	Base EV	Purified EV	Improvement
JQ.K-J.QK	-119.46	-37.75	81.71
J.QK-full	-115.63	-41.83	73.80
J.QK-J.Q.K	-96.66	-27.35	69.31
JQ.K-J.Q.K	-96.48	-28.76	67.71
JQ.K-full	-99.30	-39.13	60.17
JQ.K-JQK	-80.14	-24.50	55.65
JQ.K-JQ.K	-59.97	-8.31	51.66
J.Q.K-J.QK	-60.28	-13.97	46.31
J.Q.K-J.Q.K	-46.23	-1.86	44.37
J.Q.K-JQ.K	-44.61	-3.85	40.76
full-JQK	-43.80	-10.95	32.85
J.QK-J.QK	-96.60	-67.42	29.18
J.QK-JQK	-95.69	-67.14	28.55
full-J.QK	-52.94	-24.55	28.39
J.QK-JQ.K	-77.86	-52.62	25.23
J.Q.K-full	-68.10	-46.43	21.66
full-JQ.K	-55.52	-36.38	19.14
full-J.Q.K	-51.14	-40.32	10.82
JQK-J.QK	-282.94	-279.44	3.50
JQK-full	-273.87	-279.99	-6.12
JQK-J.Q.K	-258.29	-279.99	-21.70
J.Q.K-JQK	-156.35	-188.00	-31.65
JQK-JQK	-386.89	-433.64	-46.75
JQK-JQ.K	-274.69	-322.41	-47.72

Table 2: Effects of purification on performance of abstract strategies against an equilibrium opponent in mb/h.

appears that the improvement generally increases with the threshold whenever thresholding improves a strategy, with the biggest improvement achieved using full purification.

8. TEXAS HOLD'EM

In the 2010 Annual Computer Poker Competition, the CMU team (Ganzfried, Gilpin, and Sandholm) submitted bots that used both purification and thresholding to the two-player no-limit Texas Hold'em division. We present the results in Section 8.1. Next, in Section 8.2, we observe how varying the amount of thresholding used effects the exploitabilities of two bots submitted to the two-player limit Texas Hold'em division.

8.1 A champion no-limit Texas Hold'em program

The two-player no-limit competition consists of two subcompetitions with different scoring rules. In the *instant-runoff* scoring rule, each pair of entrants plays against each other, and the bot with the worst head-to-head record is eliminated. This procedure is continued until only a single bot remains. The other scoring rule is known as *total bankroll*. In this competition, all entrants play against each other and are ranked in order of their total profits. While both scoring metrics serve important purposes, the

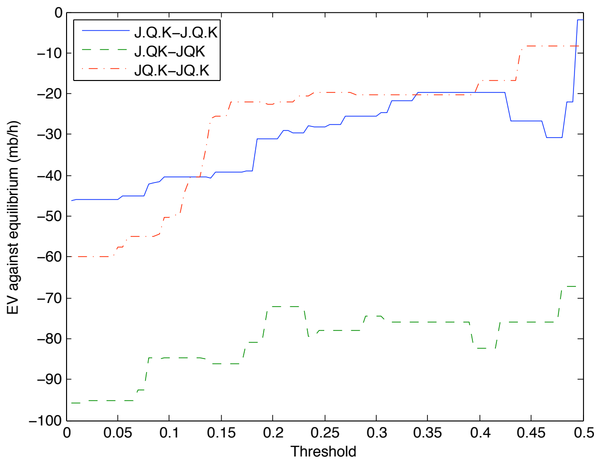


Figure 3: Effects of thresholding on performance of abstract strategies against an equilibrium opponent in mb/h.

total bankroll competition is considered by many to be more realistic, as in many real-world multiagent settings the goal of agents is to maximize total payoffs against a variety of opponents.

We submitted bots to both competitions: Tartanian4-IRO (IRO) to the instant-runoff competition and Tartanian4-TBR (TBR) to the total bankroll competition. Both bots use the same abstraction and equilibrium-finding algorithms. They differ only in their reverse-mapping algorithms: IRO uses thresholding with a threshold of 0.15 while TBR uses purification. IRO finished third in the instant-runoff competition, while TBR finished first in the total bankroll competition.

Although the bots were scored only with respect to the specific scoring rule and bots submitted to that scoring rule, all bots were actually played against each other, enabling us to compare the performances of TBR and IRO. Table 3 shows the performances of TBR and IRO against all of the bots submitted to either metric in the 2010 two-player no-limit Texas Hold'em competition.

One obvious observation is that TBR actually beat IRO when they played head-to-head (at a rate of 80 milli big blinds per hand). Furthermore, TBR performed better than IRO against every single opponent except for one (c4tw.iro). Even in the few matches that the bots lost, TBR lost at a lower rate than IRO. Thus, even though TBR uses less randomization and is perhaps more exploitable in the full game, the opponents submitted to the competition were either not trying or not able to find successful exploitations. Additionally, TBR would have still won the total bankroll competition even if IRO were also submitted.

These results show that purification can in fact yield a big gain over thresholding (with a lower threshold) even against a wide variety of realistic opponents in very large games.

8.2 Assessing worst-case exploitability in limit Texas Hold'em

Despite the performance gains we have seen from purification and thresholding, it is possible that these gains come at the expense of worst-case exploitability (see Section 4.2). Exploitabilities for several variants of a bot we submitted to the two-player limit division of the 2010 ACPC (GS6.iro) are given in Table 4; the exploitabilities were computed in the full unabstracted game using a recently developed approach [7].

Interestingly, using no rounding at all produced the most exploitable bot, while the least exploitable bot used an *intermediate* threshold of 0.15. There is a natural explanation for this seemingly surprising phenomenon. If there is too much thresholding, the resulting strategy does not have enough randomization, so it signals too much to the opponent about the agent's private information. On the other hand, if there is too little thresholding, the strategy is overfit to the particular abstraction.

Hyperborean.iro was submitted by the University of Alberta to the competition; exploitabilities of its variants are shown as well. Hyperborean's exploitabilities increased monotonically with the threshold, with no rounding producing the least exploitable bot.

Threshold	Exploitability of GS6	Exploitability of Hyperborean
None	463.591	**235.209**
0.05	326.119	243.705
0.15	**318.465**	258.53
0.25	335.048	277.841
Purified	349.873	437.242

Table 4: Worst-case exploitabilities of several strategies in two-player limit Texas Hold'em. Results are in milli big blinds per hand. Bolded values indicate the lowest exploitability achieved for each strategy.

These results show that it can be hard to predict the relationship between the amount of rounding and the worst-case exploitability, and that it may depend heavily on the abstraction and/or equilibrium-finding algorithm used. While exploitabilities for Hyperborean are more in line with what one might intuitively expect, results from GS6 show that the minimum exploitability can actually be produced by an intermediate threshold value. One possible explanation of this difference is that thresholding and purification help more when coarser abstractions (i.e., smaller abstract games relative to the full game) are used, while in finer-grained abstractions, they may not help as much, and may even hurt performance.[1] The fact that the exploitability of Hyperborean is smaller than that of GS6 suggests that it was computed using a finer-grained abstraction.

9. CONCLUSIONS

We presented two new reverse-mapping algorithms for large games: purification and thresholding. One can view these approaches as ways of achieving robustness against one's own lossy abstraction. From a theoretical perspective, we proved that it is possible for each of these algorithms to help (or hurt) arbitrarily over the standard approach, and that each can perform arbitrarily better than

[1]It is worth noting that purification and thresholding cannot help us against an equilibrium strategy if the abstraction is lossless; but even if it is lossless the algorithms may still help against actual (non-equilibrium) opponents.

	c4tw.iro	c4tw.tbr	Hyperborean.iro	Hyperborean.tbr	PokerBotSLO	SartreNL	IRO	TBR
IRO	5334 ± 109	8431 ± 156	-248 ± 49	-364 ± 42	108 ± 46	-42 ± 38		-80 ± 23
TBR	4754 ± 107	8669 ± 168	-122 ± 38	-220 ± 39	159 ± 40	13 ± 33	80 ± 23	

Table 3: Results from the 2010 Annual Computer Poker Competition for two-player no limit Texas Hold'em. Values are in milli big blinds per hand (from the row player's perspective) with 95% confidence intervals shown. IRO and TBR both use the same abstraction and equilibrium-finding algorithms. The only difference is that IRO uses thresholding with a threshold of 0.15 while TBR uses purification.

the other. However, in practice both purification and thresholding seem to consistently help over a wide variety of domains.

Our experiments on random matrix games show that, perhaps surprisingly, purification helps even when random abstractions are used. Our experiments on Leduc Hold'em show that purification leads to improvements on most abstractions, especially as the abstractions become more sophisticated. Additionally, we saw that thresholding generally helps as well, and its performance improves overall as the threshold cutoff increases, with optimal performance usually achieved at full purification. We also saw that purification outperformed thresholding with a lower threshold cutoff in the Annual Computer Poker Competition against a wide variety of realistic opponents. In particular, our bot that won the 2010 two-player no-limit Texas Hold'em bankroll competition used purification. Finally, we saw that these performance gains do not necessarily come at the expense of worst-case exploitability, and that intermediate threshold values can actually produce the lowest exploitability. There is a natural explanation for this seemingly surprising phenomenon. If there is too much thresholding, the resulting strategy does not have enough randomization, so it signals too much to the opponent about the agent's private information. On the other hand, if there is too little thresholding, the strategy is overfit to the particular abstraction.

10. FUTURE RESEARCH

Our results open up many interesting avenues for future work. In Section 6, we presented several concrete theoretical open problems related to understanding the performance of purification in random matrix games. In particular, larger games (with different degrees of abstraction) should be studied, and perhaps general theorems can be proven to augment our (statistically significant) empirical findings.

Future work should also investigate possible deeper connections between purification, abstraction, and overfitting from a learning-theoretic perspective. Is there a formal sense in which purification and thresholding help diminish the effects of overfitting strategies to a particular abstraction? Is such overfitting more prone to occur with coarser abstractions, or with some abstraction algorithms more than others? Perhaps the results also depend crucially on the equilibrium-finding algorithm used (especially for games with many equilibria). A better understanding of these phenomena could have significant practical and theoretical implications.

In addition, note that purification/thresholding is just one family of modifications to the current abstraction/equilibrium paradigm. Many other approaches are possible; for example, rounding probabilities to intermediate values (rather than to 0), or randomizing equally between the k highest-probability actions.

11. REFERENCES

[1] D. Billings, N. Burch, A. Davidson, R. Holte, J. Schaeffer, T. Schauenberg, and D. Szafron. Approximating game-theoretic optimal strategies for full-scale poker. In *IJCAI*, 2003.

[2] A. Gilpin and T. Sandholm. A competitive Texas Hold'em poker player via automated abstraction and real-time equilibrium computation. In *AAAI*, 2006.

[3] A. Gilpin and T. Sandholm. Lossless abstraction of imperfect information games. *Journal of the ACM*, 54(5), 2007.

[4] A. Gilpin and T. Sandholm. Expectation-based versus potential-aware automated abstraction in imperfect information games: An experimental comparison using poker. In *AAAI*, 2008. Short paper.

[5] A. Gilpin, T. Sandholm, and T. B. Sørensen. A heads-up no-limit Texas Hold'em poker player: Discretized betting models and automatically generated equilibrium-finding programs. In *AAMAS*, 2008.

[6] S. Hoda, A. Gilpin, J. Peña, and T. Sandholm. Smoothing techniques for computing Nash equilibria of sequential games. *Mathematics of Operations Research*, 35(2), 2010.

[7] M. Johanson, K. Waugh, M. Bowling, and M. Zinkevich. Accelerating best response calculation in large extensive games. In *IJCAI*, 2011.

[8] D. Koller, N. Megiddo, and B. von Stengel. Fast algorithms for finding randomized strategies in game trees. In *STOC*, 1994.

[9] R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a Nash equilibrium. *Games and Economic Behavior*, 63(2), 2008.

[10] D. Schnizlein, M. Bowling, and D. Szafron. Probabilistic state translation in extensive games with large action sets. In *IJCAI*, 2009.

[11] J. Shi and M. Littman. Abstraction methods for game theoretic poker. In *CG '00: Revised Papers from the International Conference on Computers and Games*. Springer-Verlag, 2002.

[12] K. Waugh, D. Schnizlein, M. Bowling, and D. Szafron. Abstraction pathologies in extensive games. In *AAMAS*, 2009.

[13] M. Zinkevich, M. Bowling, M. Johanson, and C. Piccione. Regret minimization in games with incomplete information. In *NIPS*, 2007.

Solving Non-Zero Sum Multiagent Network Flow Security Games with Attack Costs

Steven Okamoto
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
sokamoto@cs.cmu.edu

Noam Hazon
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
noamh@cs.cmu.edu

Katia Sycara
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
katia@cs.cmu.edu

ABSTRACT

Moving assets through a transportation network is a crucial challenge in hostile environments such as future battlefields where malicious adversaries have strong incentives to attack vulnerable patrols and supply convoys. Intelligent agents must balance network costs with the harm that can be inflicted by adversaries who are in turn acting rationally to maximize harm while trading off against their own costs to attack. Furthermore, agents must choose their strategies even without full knowledge of their adversaries' capabilities, costs, or incentives.

In this paper we model this problem as a non-zero sum game between two players, a sender who chooses flows through the network and an adversary who chooses attacks on the network. We advance the state of the art by: (1) moving beyond the zero-sum games previously considered to non-zero sum games where the adversary incurs attack costs that are not incorporated into the payoff of the sender; (2) introducing a refinement of the Stackelberg equilibrium that is more appropriate to network security games than previous solution concepts; and (3) using Bayesian games where the sender is uncertain of the capabilities, payoffs, and costs of the adversary. We provide polynomial time algorithms for finding equilibria in each of these cases. We also show how our approach can be applied to games where there are multiple adversaries.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, Multiagent systems*; J.4 [**Social and Behaviorial Sciences**]: [Economics]

General Terms

Theory, Security, Economics

Keywords

network security game, communication security, multiagent communication

1. INTRODUCTION

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

Many multiagent applications must utilize networks in inherently hostile environments where adversaries have strong incentives to disrupt operations, as when enemies attack vulnerable patrols and supply convoys in transportation networks using asymmetric warfare techniques. Any multiagent deployment in these environments must address the crucial issue of strategically moving assets in a secure and effective manner in the presence of such malicious adversaries. Game theory offers a natural and rigorous framework for reasoning strategically in these kinds of adversarial domains.

Such hostile network environments share six key characteristics: **(1)** The topology of the network creates exponential sized strategy spaces that cannot be solved efficiently using standard normal form techniques. **(2)** Security is not the sole criterion but must be balanced with competing performance objectives. For example, shorter paths are preferred by supply convoys to minimize fuel costs and by sensor networks to conserve battery power and reduce latency. **(3)** Patterns of behavior may be learned by the adversary. For example, the adversary may observe supply convoys in secret before planning and executing his attack. **(4)** Adversaries are rational agents who balance the harm that they can inflict with costs of attacking. **(5)** Information on the adversary's capabilities, payoffs, and costs is rarely available, and estimates must be used instead. **(6)** Multiple adversaries with differing abilities may be present.

Work in network security games has addressed the first [15, 12, 8], second [12], and third [15, 8] of these, but has left the others largely untouched. Attack costs have largely been ignored by assuming a zero-sum payoff structure, so that the incentives of the sender and adversary are exactly opposed. In general this leads to a computationally simpler problem, but does not reflect the fact that the attack costs do not factor in to the payoff of the sender. Abandoning the zero-sum assumption is also essential to addressing the fourth point because it is known that in zero-sum games it does not matter if the adversary can observe the behavior patterns before choosing a strategy [17]. In this paper we address each of these characteristics, starting most importantly by allowing non-zero sum payoffs based on attack costs.

We model the problem as a game between a sender and an adversary. The sender chooses a flow through the network from the source nodes to the sink. The adversary chooses one or more attacks from a set of possible attacks, where each attack adds penalties to one or more link in the network. For example, one attack may be to jam a node in a communication network, thereby interfering with the communication with that targeted node and also (although to a lesser extent) to all communication with neighboring nodes as well. When the sender utilizes a link that has been attacked, he suffers harm proportional to the total penalty on the link and the amount of flow being sent on the link. The adversary incurs a cost for each attack, and different attacks may have different costs reflecting the differing degrees of difficulty or ease of attack. The sender seeks to minimize harm while the attacker seeks to maximize the harm minus the attack costs.

In this paper we advance the state of the art by combining the existing approaches with three major new contributions. First, we assume non-zero sum payoffs due to attack costs that adversary incurs in attacking the network. These costs factor into his payoff but not the payoff of the sender. We provide polynomial time algorithms based on linear programs (LPs) for finding Nash equilibria in these games. Second, the non-zero sum payoffs allow the possibility of the sender improving his payoff by committing to strategies in a Stackelberg game. We show that the existing solution concepts are inappropriate for network security games and introduce a refinement of the Stackelberg equilibrium based on the sender's ability to affect the adversary's strategy by deviating from equilibrium behavior. Finally, we consider games of incomplete information where the sender knows only probability distributions over the maximum number of nodes that the adversary can attack, and the adversary's payoffs and costs. We formulate these as Bayesian games and provide polynomial time algorithms for finding equilibria. We also show how this approach can be generalized to model games with multiple adversaries.

2. SIMULTANEOUS GAME

2.1 Model

We start with the network flow security game with attack costs but no uncertainty. This game is played between a sender and an adversary taking actions on a network represented by a directed graph $G = (V, E)$ with $n = |V|$ nodes and $m = |E|$ edges. The sender chooses how to send flow from a set $S \subset V$ of *source nodes* to the sink node $t \in V$. The amount of flow originating at a node $v \in V$ that must be sent to t is denoted by b_v, with $b_v > 0$ for all $v \in S$ and $b_v = 0$ for all $v \notin S$. The sender's strategy space $\mathcal{F}$ is the set of all feasible flows from the source nodes to the sink. Flows may be divided on alternate paths from the source nodes to the sink[1], leading to a continuous strategy space for the sender if there are at least two paths from any source node to the sink. A sender strategy f is represented as a $m \times 1$ vector where f_e is the amount of flow sent on edge $e \in E$. For convenience, for an edge $(u, v) \in E$, $f_{(u,v)}$ is denoted simply as f_{uv}.

The adversary chooses attacks from a set A. The adversary has a cost for each attack, represented by the $|A| \times 1$ attack cost vector c, where $c_a \geq 0$ is the the cost suffered by the adversary for attack $a \in A$. The adversary can execute up to k attacks simultaneously. Thus the adversary's set of pure strategies $\mathcal{A}$ is the set of all subsets of A of size at most k, which has size $\Theta(|A|^k)$, and his set of mixed strategies is the set of all probability distributions over $\mathcal{A}$. Instead of representing mixed strategies explicitly, we use the marginal probability distribution represented by the $1 \times |A|$ vector p, where p_a is the marginal probability of the adversary executing $a \in A$. This is sufficient for computing payoffs (and hence equilibrium behavior) [12], and so we sometimes refer to p as the adversary's mixed strategy. Because of the size of $\mathcal{A}$, even describing a mixed strategy explicitly requires exponential time in general, but it is possible to efficiently sample a pure strategy in conformance with p using algorithms such as comb sampling [15] or weighted random sampling [5].

The payoff for the sender in the game is quantified by the *harm* suffered as a result of the adversary's attacks. The harm is represented by a *harm matrix* M with $|A|$ rows and m columns, where each row specifies the penalties on edges caused by an attack so that entry M_{ij} is the per-unit-flow harm suffered when the adversary executes attack a_i and and the sender transmits flow on edge e_j. Harm for multiple attacks is summed, as occurs when a convoy must endure multiple attacks on its route, or when multiple jamming attacks in different parts of an ad hoc network additively increase the latency of messages. This representation models a broad range of harm functions that cannot be represented in other network security games [12]. When the sender plays f and the adversary plays p, the total expected harm is pMf and so the sender's payoff is $-pMf$. The payoff for the adversary depends on the harm that the sender suffers and the cost of the attacks, computed as $pMf - pc$. We refer to the adversary's payoff as his *reward*.

When the players choose their actions without any observations of the other player the game is played as a simultaneous move game and we use the familiar Nash equilibrium solution concept. A strategy profile (f^*, p^*) is a Nash equilibrium if f^* is a best response to p^* (for the sender) and p^* is a best response to f^* (for the adversary). In equilibrium, neither player has incentive to deviate and hence both are indifferent between their possible strategies.

To illustrate the importance of the attack costs on the Nash equilibrium, consider the network in Figure 1. Assume that $b_s = 1$ and $k = 1$ and that the adversary can choose to attack the top path or the bottom path with harm matrix

$$M = \begin{bmatrix} 102 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \end{bmatrix}.$$

Let f_i denote the amount of flow on edge e_i and note that the sender's strategy is fully specified by f_1 as $f_2 = 1 - f_1$. Let p_1 and p_2 denote the probability of attacking the top and bottom paths respectively.

When attack costs are zero, the adversary never has incentive not to attack, so $p_1 + p_2 = 1$. In equilibrium the adversary is indifferent between the two attacks so $102f_1 = 3(1 - f_1)$ and so $f_1 = 3/105$. Similarly, the sender is indifferent between the two paths so $102p_1 = 3(1 - p_1)$ and so $p_1 = 3/105$. An intuitive interpretation is that the sender sends most of his flow on the bottom path because of the lower potential for harm, while the adversary, being able to deduce this, attacks the bottom path with high probability because that's where most of the flow is.

Now suppose that attacking the top path has a cost $c_1 = 100$. The adversary's equilibrium strategy remains the same because the sender's payoff hasn't changed, but now for the adversary to be indifferent it must be that $102f_1 - 100 = 3f_2$

[1]This approach can be used even when the asset moving through the network cannot be split, as with a convoy that must travel intact. The flow is then an efficient polynomial-sized representation for a mixed strategy over the exponential number of paths from the source nodes to the sink. Splits in the flow correspond to randomization over possible paths.

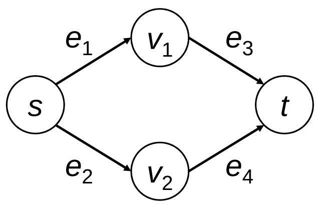


Figure 1: An example of a network with two possible paths.

so that $f_1 = 103/105$. An intuitive explanation is that the adversary attacks the top path with low probability because of the high attack cost, and the sender, deducing this, sends most of the flow on the top path despite the high potential for harm because the adversary is unlikely to attack there.

The attack costs can also cause the adversary to not execute his maximum number of attacks because the cost outweighs the harm. For example, if $c_1 > 102$ then the sender can set $f_1 = 1$ and a best response by the adversary is to choose $p_1 = p_2 = 0$.

2.2 Computing Nash Equilibrium

Finding Nash equilibria in general non-zero games is computationally more expensive than finding equilibria in zero-sum games. In addition, there may be multiple equilibria with different payoffs for both players, which can complicate the matter of choosing a strategy. In this section we show that these concerns do not arise in the network flow game with attack costs, because the Nash equilibria in this game are precisely those of the zero-sum game where both payoffs are affected by attack cost.

To prove this we will use the following lemma:

LEMMA 1. *Let p be an adversary's strategy and let f be a sender's strategy. Then f is a best response to p if and only if f minimizes the adversary's expected payoff given p.*

PROOF. We start with the forward direction. Assume f is a best response to p. Because f is a best response to p, it follows that f must minimize harm, pMf. Therefore it must also minimize $pMf + \alpha$ for any α that is constant (with respect to f). In particular, f must minimize $pMf - pc$, the adversary's expected payoff. The proof of the reverse direction is similar. $\square$

We can now prove the theorem:

THEOREM 1. *(f, p) is a Nash equilibrium for the network flow game with attack costs if and only if f minimizes the maximum adversary payoff and p maximizes the minimum adversary payoff.*

PROOF. We start with the backward direction. The fact that p maximizes the adversary's payoff given f follows directly from the assumption that p is a maximin strategy for the adversary's payoff. Thus p is a best response to f. It also follows that f minimizes reward given p because f is a minimax strategy for the adversary's payoff. Thus by Lemma 1 f is a best response to p. Therefore (f, p) are mutual best responses and hence form a Nash equilibrium.

Now, suppose that (f, p) is a Nash equilibrium. We prove that f must minimize the maximum adversary payoff by contradiction. Suppose that f is not a minimax strategy and let f' be a minimax strategy. Then there exists marginal probability vector p'' such that for all marginal probability vectors p', $p'Mf' - p'c < p''Mf - p''c$. In particular, for $p' = p$, we get

$$\begin{aligned} pMf' - pc &< p''Mf - p''c \\ &\leq pMf - pc \qquad (p \text{ is a best response to } f) \end{aligned}$$

But $pMf' - pc < pMf - pc$ implies that $pMf' < pMf$, which means that f is not a best response to p (for the sender), contradicting (f, p) being a Nash equilibrium. Hence f must be a minimax strategy.

It then follows readily that p must be a maximin strategy, as the adversary seeks to maximize reward. $\square$

Because of Theorem 1, finding an equilibrium sender strategy reduces to finding a minimax strategy. This can be found efficiently by using the linear program LP 1, despite the large strategy spaces for both players:

LP 1 Equilibrium Sender Strategy with Attack Costs
Input: G, M, c, k
Output: f, R, λ

$$\underset{f,R,\lambda}{\text{Minimize}}\ kR + \sum_{a \in A} \lambda_a \tag{1}$$

subject to:

$$R \geq \text{row}_a[M]f - c_a - \lambda_a \qquad \forall a \in A \tag{2}$$

$$\sum_{(v,u)\in E} f_{vu} = b_v + \sum_{(u,v)\in E} f_{uv} \qquad \forall v \in V \backslash \{t\} \tag{3}$$

$$f_{uv} \geq 0 \qquad \forall (u,v) \in E \tag{4}$$

$$\lambda_a \geq 0 \qquad \forall a \in A \tag{5}$$

The adversary's expected payoff is represented by R and the λ_a variables. For a flow f, the *potential reward* of an attack a is the amount of additional payoff that the adversary will get if he plays a. This is calculated as $\text{row}_a[M]f - c_a$ where "$\text{row}_a[M]$" denotes the row of M corresponding to attack a. When $k = 1$, a best response by the adversary is to play an attack with maximum potential reward. Thus $\lambda_a = 0$ for all a and thus R is the amount of reward gained and will be the maximum reward that can be gained from any single attack, as required by Equation (2). Thus the sender will minimize the maximum potential reward. When $k > 1$, the sender no longer needs to minimize the potential reward of a single attack, but rather must minimize the sum of potential rewards for a set of attacks of size k. In some cases, the sender may benefit from the adversary playing attacks with higher potential reward if it allows other attacks to have lower potential reward, thus resulting in a net decrease in total potential reward. This idea is captured by the λ_a variables, which allow an attack to "borrow" potential reward from other nodes to form a net decrease. Variable R now represents the minimum potential reward among the nodes that may be attacked by the adversary in a best response. Rewriting Equation 2 to get $R + \lambda_a \geq \text{row}_a[M]f - c_a$, we see that the reward potential for a node is the minimum plus the "borrowed" amount. Each attack a will contribute $R + \lambda_a$ reward to the total, which is shown in the objective function $kR + \sum_{a\in A} \lambda_a$. LP 1 has $|A| + m + 1$ variables and at most $2|A| + n + m - 1$ constraints. Thus it can be solved in polynomial time (with respect to n and $|A|$).

For adversary, we take the dual to the sender's LP and get the following program LP 2:

LP 2 Equilibrium Adversary Strategy with Attack Costs

Input: G, M, c, k

Output: r, p

$$\underset{r,p}{\text{Maximize}} \left(\sum_{v \in S} b_v r_v \right) - p\ c^T$$

subject to:

$$r_u \le r_v + p\ \mathrm{col}_{(u,v)}[M] \qquad \forall (u,v) \in E$$

$$r_t = 0$$

$$\sum_{a \in A} p_a \le k$$

$$0 \le p_a \le 1 \qquad \forall a \in A$$

The vector p represents the marginal probabilities of attacking nodes. The vector r encodes the sender's best response to p, with r_v being the least harm that the sender can suffer for each unit of flow sent from v to the sink. At the sink, no harm can be suffered (the flow is already at the sink). From a node u other than the sink, we observe that the sender must send flow on one of the outgoing edges (u, v) to a neighbor v. The harm suffered will be equal to the harm suffered crossing (u, v), plus the harm suffered from v to t. Thus, the *least* harm suffered sending from u to t will be equal to the minimum of the harm suffered from sending flow on (u, v) plus the least harm from v to t. That is, $r_u = \min_{(u,v) \in E} r_v + p\ \mathrm{col}_{(u,v)}[M]$ (where "$\mathrm{col}_{(u,v)}[M]$" denotes the column in M for edge (u, v)), which is captured by the constraint in Equation **??**. Because the sender needs to send b_v endogenous flow from node v to t (with $b_v = 0$ for $v \notin S$), the reward for the adversary is $(\sum_{v \in S} b_v r_v) - p\ c^T$, the objective that is maximized by LP 2.

We note that by the strong duality theorem, LP 2 finds the maximin adversary payoff strategy, despite the constraint in Equation **??** considering minimum harm (the sender's payoff), not the adversary's payoff! This phenomena is well established by Theorem 1: when the adversary optimizes his strategy against a sender who is trying to minimize harm, it is the same as optimizing the adversary strategy against a sender who is trying to minimize the adversary's payoff. That is, the sender's best response behavior in the non-zero sum game where his payoff is just based on harm and the adversary's payoff is reward is the same as the sender's best response behavior in the *zero sum* game where both players' payoffs are based on reward.

3. STACKELBERG GAME

In this section we consider the Stackelberg game in which the sender plays first, committing to a strategy. We show how two commonly used solution concepts, the strong and weak Stackelberg equilibria, are inappropriate for sequential network security games, and provide a polynomial time algorithm for finding a more nuanced equilibrium.

3.1 Model

In the previous section we described the simultaneous game where the sender and adversary act without observing each other's actions. However, in many settings this is not the case. For example, convoys in support of persistent military or humanitarian relief missions will operate over extended periods of time and the adversary can observe routes taken over time to build up an estimate of the sender's mixed strategy before choosing which attacks to launch. These types of settings are commonly modeled as *Stackelberg games*, a type of sequential game in which one player (the "leader") moves first, committing to a mixed strategy. The second player (the "follower") can then observe that mixed strategy and choose an appropriate response. It is known that in Stackelberg games the leader can sometimes improve his equilibrium payoff (and cannot decrease it, under mild assumptions) compared to his equilibrium payoff in the simultaneous move game [14].

In a two-player Stackelberg game the follower's strategy is a function that maps mixed strategies of the leader to mixed strategies of the follower. In the network flow security game with attack costs, the adversary's strategies are functions $g : \mathcal{F} \to \mathcal{A}$ that map each flow to an adversary mixed strategy. Let $\mathcal{G}$ denote the set of all such functions. A Stackelberg equilibrium (f^*, g^*) is a refinement of subgame perfect Nash equilibrium where (f^*, g^*) are mutual best responses, i.e.,

$$g^*(f^*)Mf^* = \min_{f \in \mathcal{F}} g^*(f)Mf$$

$$g^*(f^*)Mf^* - g^*(f^*)c = \max_{g \in \mathcal{G}} g(f^*)Mf^* - g(f^*)c.$$

both hold, and $g^*(f)$ is a best response to f for all $f \in \mathcal{F}$ (the follower always plays optimally, even off the equilibrium path). Computing a best response function g for the adversary is straightforward: given f, greedily choose up to k attacks that have maximum payoff to the adversary, excluding any that would contribute negative payoff because the attack cost is too high. Note that there will be multiple best response functions if there is some f for which the set of k attacks yielding highest adversary payoff is not unique, and that these best response functions may yield different payoffs to the sender because of heterogeneous attack costs. Thus there may be multiple Stackelberg equilibria that have the same sender strategy but different sender payoffs.

Traditionally two kinds of Stackelberg equilibrium are distinguished: strong Stackelberg equilibrium (SSE), where the follower's best response function always maps to a strategy that maximizes the leader's payoff; and weak Stackelberg equilibrium (WSE), where the follower's best response function always maps to a strategy that minimizes the leader's payoff [9]. The pessimistic WSE is the more natural solution concept for security applications, which tend to focus on worst case behavior. Despite this, SSE, which assumes that the malicious adversary breaks ties in the leader's favor, has been considered more often in the literature for two technical reasons: (1) a SSE is guaranteed to exist in every Stackelberg game, while a WSE may not; and (2) it is often claimed that the leader can *induce* the adversary to play the desired best-case strategy by deviating by an arbitrarily small amount from the equilibrium in order to break the adversary's indifference [14]. We will show that both of these arguments are inappropriate for the network security game, but first illustrate several important concepts by example.

Recall the example in Figure 1 with $c_1 = 100$. The adversary is indifferent when $f_1 = 103/105$, prefers the top path when it is $f_1 > 103/105$, and prefers the bottom path when $f_1 < 103/105$. Thus all best response functions $g_1 : [0, 1] \to [0, 1]$ mapping f_1 to the probability of attacking

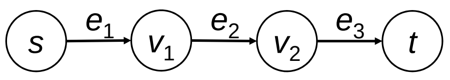


Figure 2: A network topology in which the sender cannot induce a strong Stackelberg equilibrium.

the top path must satisfy $g_1(f_1) = 0$ when $f_1 < 103/105$ and $g_1(f_1) = 1$ when $f_1 > 103/105$, and any value $g_1(f_1) \in [0, 1]$ is acceptable for $f_1 = 103/105$.

In the unique simultaneous Nash equilibrium, $p_1 = 3/105$, so that the sender was indifferent between the top and bottom paths but sent $f_1 = 103/105$ flow on the top path and $f_2 = 2/105$ flow on the bottom path, suffering harm on both paths. It follows that $f_1 = 103/105$ is a best response to the adversary's best response function g_1^{NE} with $g_1^{NE}(103/105) = 3/105$. Thus the simultaneous Nash equilibrium naturally gives rise to a Stackelberg equilibrium strategy, with the same payoff to the sender as in the Nash equilibrium, $-306/105$. In the SSE the adversary attacks the bottom path (i.e., $g_1^{SSE}(103/105) = 0$), resulting in a much higher payoff to the sender, $-6/105$. It is easy to see that there are no other Stackelberg equilibria for this game. For example, there is no WSE because if the adversary played the worst-case best response with $g_1^{worst}(103/105) = 1$, then the sender would have incentive to deviate by decreasing f_1.

The sender's strategy is the same in both of these equilibria which means that his payoff ultimately depends on the choice of the indifferent adversary. However, note that the sender can deviate slightly from his equilibrium strategy by playing $f_1 = 103/105 - \varepsilon$ for some small $\varepsilon > 0$, in order to incentivize the adversary to attack the bottom path. By doing this the sender will receive a payoff of $-(6/105 + 3\varepsilon)$ instead of the $-6/105$ that he would earn in the SSE, but as ε is made arbitrarily small his strategy converges to the SSE strategy.

It is not always possible to induce the SSE by deviating from an equilibrium strategy. Consider the network in Figure 2, and assume that A contains two attacks, one that affects e_1 and one that affects e_2, with harm matrix

$$M = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{bmatrix},$$

and costs $c_1 = 3$ and $c_2 = 1$. The sender has no choice as his only pure strategy is to send the full flow on the single path from s to t. At the same time, the adversary is indifferent to the choice of attack as they both yield him a payoff of 2 and so might choose either of them.

3.2 Inducing Locally Optimal Equilibria

Because the WSE may not exist and the SSE may not be attainable, we address the problem of how the sender can deviate from a Stackelberg equilibrium strategy f to induce a Stackelberg equilibrium (f, g) that yields him maximum payoff. We call this a locally optimal inducible Stackelberg equilibrium (loptISE). It is locally optimal because the value of the Stackelberg equilibrium that is induced depends on the starting equilibrium strategy f. The starting strategy that we use is one that arises naturally from the simultaneous game Nash equilibrium strategy found by LP 1, which we now show to always be a Stackelberg equilibrium.

LEMMA 2. *If a strategy profile (f, p) is a Nash equilibrium for the network flow security game with attack costs found by LP 1 then (f, g) is a Stackelberg equilibrium for the Stackelberg network flow security game with attack costs for a best response function g with $g(f) = p$.*

PROOF. We construct g as a best response function with $g(f) = p$. For $f' \neq f$, we set $g(f')$ to be the best response that maximizes $g(f')Mf'$. By definition of Nash equilibrium, f maximizes $g(f)Mf$. Because LP 1 finds a minimax strategy, it follows that $g(f')Mf' \geq g(f)Mf$ for all $f' \in \mathcal{F}$. Thus, f is a best response to g in the Stackelberg game, and so (f, g) is a Stackelberg equilibrium. □

Algorithm 1 Computing deviation to find optimal inducible Stackelberg equilibrium

```
1: Find Nash equilibrium flow $f$ using LP1
2: Set $A'$ to be the set of minimum adversary payoff candidate attacks.
3: Set $k' \leq k$ to be the number of candidate attacks that must be chosen from $A'$.
4: if $|A'| \leq k'$ then
5:    Return.
6: Add dummy source $s_0$ to $G$. Set $I' \leftarrow \emptyset$. Set $f^\varepsilon$ to be the empty flow.
7: while $|I| < k'$ do
8:    Set $F \leftarrow \emptyset$.
9:    for all $a \in A'$ do
10:       Solve $(f^a, H, H') \leftarrow$ LP3$(G, M, A', a)$
11:       if $H - H' \geq 0$ then
12:          $F \leftarrow F \cup \{f^a\}$
13:    Set $Y \leftarrow \{a | f^a \in F$ with minimum row$_a[M]f^a\}$.
14:    Set $Z \leftarrow \{a | \exists a' \in Y$ s.t. row$_a[M]f^a =$ row$_{a'}[M]f^{a'}\}$
15:    Set $A' \leftarrow A' \backslash Z$ and $I \leftarrow I \cup Z$.
16:    Set $f^\varepsilon \leftarrow f^\varepsilon + \sum_{a \in Y} f^a$
17: Set $f \leftarrow f + \varepsilon f^\varepsilon$
18: for all $s \in S$ do
19:    Normalize outgoing flow.
```

Algorithm 1 computes a deviation from an equilibrium strategy that the sender can use to induce a loptISE. We first sketch the high level approach before delving into the details. The sender starts with a Nash equilibrium flow f (which is also a Stackelberg equilibrium strategy according to Lemma 2), then computes the set A' of *candidate attacks* that might be chosen by the adversary as part of a best response to f and that the adversary is indifferent between. The sender tries to incentivize the adversary to choose certain of these candidate attacks by adding small amounts of flow. Intuitively this approach exploits what we observed in the example: parallel paths allow the sender freedom to deviate and bias the adversary's choice toward less harmful attacks, while a sequential topology does not permit this flexibility and instead the adversary will be assumed to choose the most harmful attack (a worst-case approach to security). However, the process is not as obvious when dealing with general attacks, each of which may affect an arbitrary set of links with heterogeneous harm values. Instead of choosing a simple path, the sender tries to find a flow for each candidate attack that will cause the sender to prefer to play that attack over all other candidate attacks. When presented with multiple options, the sender chooses one that causes the least harm (i.e., increases his payoff the most). The repeats until the sender has incentivized all of the adversary's attacks, to less harmful attacks when possible and to most harmful attacks otherwise. The deviation flows (which may be made arbitrarily small) are then superimposed on the original flow to generate the desired deviation.

Computing the candidate attacks is straightforward. Given f, we compute the payoff that the adversary would receive for each attack $a \in A$ as $\rho_a = \text{row}_a[M]f$. The adversary will choose up to k attacks with the highest ρ_a, and will only choose attacks with $\rho_a \geq 0$. If there are more than k attacks with $\rho_a \geq 0$, we compute the multiset of attacks with the highest k values of ρ_a (with repetition). For example, if the multiset of ρ_a values is $\{10, 10, 8, 7, 7, 7, 0, -2\}$ and $k = 4$, then there are 6 candidate attacks, $\{a \in A | \rho_a \geq 7\}$. The adversary's best response will always choose the attacks with ρ_a strictly greater than the minimum, so we need only consider A' to be those with minimum ρ_a values. In the previous example, that would mean that the best response always plays the two attacks with $\rho_a = 10$ and the one attack with $\rho_a = 8$ (i.e., these cannot be affected by the sender), so we are left to choose $k' = k - 3 = 1$ candidate attacks from among the three remaining with $\rho_a = 7$.

A dummy source node s_0 is added to G and connected to each source in $s \in S$, to allow deviant flows from any source. The set of induced attacks I, is initialized as empty. The overall deviation flow f^ε is initially empty.

The algorithm then iterates up to k' times in the loop starting at line 7. On each iteration it attempts to greedily induce the adversary to choose attacks that maximally increase the sender's payoff. The set of best deviant flows is F. For each $a \in A'$, we compute the deviation f^a that makes the adversary prefer to play $a \in A'$ over other candidate attacks. This is achieved by LP 3:

LP 3 Stackelberg deviating flow for a.

Input: G, M, A', a
Output: f^a, H, H'

$$\underset{f^a, H, H'}{\text{Maximize}} \; H - H'$$

subject to:

$$H \leq \text{row}_a[M]f^a$$

$$H' \geq \text{row}_{a'}[M]f^a \qquad \forall a' \in A' \backslash \{a\}$$

$$\sum_{(v,u)\in E} f^a_{vu} = \sum_{(u,v)\in E} f^a_{uv} \qquad \forall v \in V \backslash \{s_0, t\}$$

$$\sum_{(s_0,u)\in E} f^a_{s_0 u} = 1$$

$$f^a_{uv} \geq 0 \qquad \forall (u,v) \in E$$

Variable H represents the amount of harm inflicted by attack a, while H' is the amount of harm inflicted for any other attack $a' \in A'$ with $a' \neq a$. LP 3 seeks to maximize the difference $H - H'$. By assumption the adversary is indifferent between candidate attacks so costs can be ignored; relative changes in reward are solely due to relative changes in harm. Thus if $H - H' \geq 0$ the adversary receives at least as much reward by choosing a and so the sender can induce the adversary to play a (perhaps with other attacks), while if $H - H' < 0$, the sender cannot yet induce the adversary to prefer a over other candidate attacks.

Of the attacks that can be induced on this iteration, the sender chooses those that cause minimum increase in harm. These may not be unique (i.e., when $H - H' = 0$ for multiple a), so Y is the set of all such candidate attacks with minimum increase in harm that the adversary can be induced to attack on this iteration. The deviation flows that are used to induce these attacks may also induce other attacks (which have the same increase in harm), so the set Z contains all the attacks that will be induced in the current iteration. These are removed from the candidate attacks and added to the induced attacks in line 15, and the deviation flows for this iteration are superimposed on the total deviation flow f^ε before starting a new iteration.

The loop terminates when the requisite number of attacks have been induced. On each iteration of the loop, I grows and A' shrinks. It is not possible for A' to become empty prior to the termination of the loop. Prior to beginning the loop, $|A'| > k'$ (lines 4 – 5) and on every iteration the same number of attacks are added to I as are removed from A'. Thus, $|I| \geq k'$ no later than the iteration when $A' = \emptyset$.

In line 17 the deviation flow is scaled and superimposed on the equilibrium flow, and in line 19 the amount of flow (which increased due to the addition of the deviation flow) is normalized at each source node so that the total amount of flow is maintained with the addition of the deviation.

THEOREM 2. *Algorithm 1 runs in time polynomial in the size of G and A.*

PROOF. Each line in the algorithm can clearly be executed in polynomial time. The for loops in lines 9 – 12 and lines 18 – 19 iterate at most $\Theta(|A|)$ and $\Theta(n)$ time, respectively. In each iteration of the main loop from lines 7 – 16 at least 1 attack is added to $|I|$ and therefore the loop cannot iterate more than $|I| = \Theta(|A|)$ times. □

4. BAYESIAN GAMES

In many security applications, it is unrealistic to assume that complete information on the adversary is available because adversaries are hostile and usually secretive. In military and law enforcement domains, intelligence analysts, criminologists, and other experts collect relevant data on real and possible adversaries and develop inherently uncertain estimates of their capabilities and motives. In this section we represent that uncertainty as probability distributions over the maximum number of attacks that they can execute, the harm matrices, and the attack costs. We develop ways to reason strategically over this incomplete information by adopting the Bayesian game framework and find polynomial time algorithms for finding equilibria.

4.1 Uncertain k

In many situations it is not possible for the sender to know the adversary's capabilities with certainty. The sender can act as if he has the full knowledge, but he then might perform badly. For example, suppose that the game is the same as in Figure 1, but now $k = 2$. In equilibrium, the adversary strategy is $p_1 = 1/34$ and $p_2 = 1$, and the sender strategy is $f_1 = 100/102$ and $f_2 = 2/102$. The expected harm for the sender will thus be 3. However, if the sender does not know that $k = 2$ now, and continue to play his strategy for $k = 1$, the adversary will exploit it and will always attack v_1 and v_2. The sender's harm will thus increase to 100.116. Therefore, when the sender is not sure about the exact value of k, he will have to estimate it. We represent this by a probability distribution q over possible values of k, which we assume is known to both players. Given this distribution, we formulate the sender's problem as a *Bayesian game.* A Bayesian game is one in which information about characteristics of the other players is incomplete. There is a probability distribution over possible *types* for each player, and the type of a player determines that player's payoff function. In our case, the sender has only one type, and the type of the adversary is determined by the value of k. We denote the probability that the adversary is of type k as q_k. The sender's optimal equilibrium strategy can be computed using the following linear program:

LP 4 Equilibrium Sender Strategy in a Bayesian game (uncertain k)

Input: G, M, A, c, k, q
Output: f, $\{R^k\}$, $\{\lambda^k\}$

$$\underset{f,\{R^k\},\{\lambda^k\}}{\text{Minimize}} \sum_{k=1}^{|A|} q_k \left(kR^k + \sum_{a\in A} \lambda_a^k \right)$$

subject to:

$$R^k \geq \text{row}_a[M]f - c_a - \lambda_a^k \qquad \forall a \in A, \forall k \in [1..|A|]$$
$$\sum_{(v,u)\in E} f_{vu} = b_v + \sum_{(u,v)\in E} f_{uv} \qquad \forall v \in V \backslash \{t\}$$
$$f_{uv} \geq 0 \qquad \forall (u,v) \in E$$
$$\lambda_a^k \geq 0 \qquad \forall a \in A, \forall k \in [1..|A|]$$
$$R^k \geq 0 \qquad \forall k \in [1..|A|]$$

This LP is similar to LP 1, except that instead of minimizing maximum adversary expected payoff for a specific value of k, it minimizes the weighted sum of the expected rewards over possible values of k, weighted by their probability. The number of variables and constraints is still polynomial in n and $|A|$ and so this LP can be solved in polynomial time.

We must verify that in the Bayesian game, minimizing the adversary's maximum expected payoff also maximizes the sender's expected payoff.

THEOREM 3. $(f, \{p^k\})$ *is a Nash equilibrium for the nonzero sum game if and only if f minimizes the maximum expected adversary payoff and $\{p^k\}$ maximizes the minimum expected adversary payoff.*

The proof is similar to the proof of Theorem 1, and we omit it due to space constraints.

Even though the adversary knows his type (i.e., the correct value of k) he cannot use LP 2 to find his equilibrium strategy since the sender does not know the exact value of k. Instead, we can take the dual to the sender's LP. Due to space constraints we omit the exact description.

5. UNCERTAIN PAYOFFS

Another way in which the sender may be uncertain of the adversary is by not knowing the payoffs and costs. Suppose instead that he has a probability distribution r over possible l payoff matrices and attack costs (types of adversaries). For $i \in [1..l]$, let r_i be the probability that the adversary is of type i with a harm matrix M^i and cost of attacks c^i.

LP 5 Equilibrium Sender Strategy in a Bayesian game (uncertain M and c)

Input: G, M, c, k, r
Output: f, $\{R^i\}$, $\{\lambda^i\}$

$$\underset{f,\{R^i\},\{\lambda^i\}}{\text{Minimize}} \sum_{i=1}^{l} r_i \left(kR^i + \sum_{a\in A} \lambda_a^i \right)$$

subject to:

$$R^i \geq \text{row}_a[M^i]f - c_a^i - \lambda_a^i \qquad \forall a \in A, \forall i \in [1..l]$$
$$\sum_{(v,u)\in E} f_{vu} = b_v + \sum_{(u,v)\in E} f_{uv} \qquad \forall v \in V \backslash \{t\}$$
$$f_{uv} \geq 0 \qquad \forall (u,v) \in E$$
$$\lambda_a^i \geq 0 \qquad \forall a \in A, \forall i \in [1..l]$$

The following linear program computes the sender's optimal equilibrium strategy:

As before, the adversary's equilibrium strategy can be computed by taking the dual of LP 5.

If we assign $r_i = 1$ for every $i \in [1..l]$ we get a linear program which solves another interesting variant of our problem. Consider a game with one sender and multiple adversaries. The adversaries choose their strategies independently of each other (i.e., no colluding). The adversaries have different harm matrices and costs for attacking nodes and the total harm to the sender is the sum of the harm resulting from each adversary's attack. The payoff to each adversary depends only on his own strategy and the sender's strategy; it does not depend on the strategies of any of the other adversaries. For now, let's assume that every adversary can attack k nodes. By assigning $r_i = 1$ for every $i \in [1..l]$ we get that LP4 finds the optimal equilibrium strategy for the sender in the multiple adversaries game too! As for the adversary's equilibrium strategies we get an interesting observation: since the strategies can be computed by the dual of LP4, they are in fact correlated. Even though the adversaries choose their strategies independently of each other, due to the strategic consideration they behave as if they coordinate their moves.

6. RELATED WORK

Problems similar to the one we address in this paper have been studied in operations research [16, 7], robotics [6, 2], and multiagent systems [15, 8]. Many of these have also taken the perspective of the player who selects nodes or edges in the network to impair the other player who chooses paths through the network. The study of network interdiction [16, 7] looks at problems where an interdictor chooses edges or nodes to damage or destroy destroy ("interdict") in order to impair the ability of an enemy moving through the network, for example for by forcing it to take longer paths [7]. An early study of single source, single sink zero-sum games where the interdictor interdicts a single edge found that the equilibrium strategy is to only interdict edges in the minimum cut [16]. Similar results were found in network routing settings [3], and more recently in games where multiple edges can be interdicted[15, 8]. In evader-pursuer games [6, 2], both players move through the network. In path disruption games [1] multiple cooperative agents work together to interdict an adversary, in contrast to our setting where both sides are assumed to be monolithic players.

In most of these related problems, the payoff depends on the probability that at least one attack occurs on a pathway; multiple attacks on the same pathway either are not possible or incur no additional penalty. This models situations like placing checkpoints to intercept the sender; once caught, the sender cannot be caught again. In contrast, in our problem the same pathway may be subject to multiple attacks or a single attack may affect multiple edges on the same pathway, resulting in additional harm. This is useful for settings where the sender continues after an attack, as when convoys fight their way through ambushes or robots clear obstacles. Games with similar payoffs have been solved in the context of communication networks using linear programming [12] and Markov Decision Processes using oracle algorithms [11]. However, these approaches have assumed zero-sum games.

Stackelberg games [13, 15] have recently been a common framework for security where patterns of behavior may be observed and learned by the adversary, as opposed to more traditional simultaneous games[3, 10]. Stackelberg games generally allow the leader to find equilibrium strategies with higher payoff than in a simultaneous game, but only in non-zero sum games [17, 14]. Computing the optimal strategies to commit to is solvable in polynomial time in the normal form game [4], but this is not practical in our games which have exponential-sized strategy spaces. The traditional solution concept considered in all of these is the strong Stackelberg equilibrium, which is questionable for the worst-case reasoning common in security settings and is not appropriate for the network security games we consider.

7. CONCLUSIONS AND FUTURE WORK

In this paper we considered non-zero sum network security games where the adversary incurs costs to attack the network. We proved that the equilibria in this non-zero sum game correspond exactly to the equilibria in a related zero-sum game, and used this insight to develop linear programs (LPs) to find the equilibrium strategies. While the strategies were the same as in the zero-sum game, the payoffs were not, which allowed the sender to benefit by committing. We introduced a new Stackelberg equilibrium, the locally optimal inducible Stackelberg equilibrium (loptISE) that is particulary well suited for network security games, and provided a polynomial time algorithm for calculating the way in which the sender can deviate from an equilibrium strategy to get achieve strategy profiles arbitrarily close to the loptISE. We also found LPs to solve for equilibria in Bayesian games where the sender is uncertain of capabilities, payoffs, and costs of the adversary he faces.

In future work we seek to extend the Stackelberg framework to our Bayesian games. Commitment is computationally more difficult in Bayesian normal form games, so it will be interesting to see if we can further leverage the payoff and network structure to find polynomial time algorithms. We will also try to extend our results on loptISE to a globally optimal equilibrium, which seems possible given the relationships between the simultaneous and Stackelberg equilibria in the network flow security game.

Acknowledgments

This research has been sponsored in part by AFOSR MURI award number FA95500810356 and in part by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-06-3-0001.

8. REFERENCES

[1] Y. Bachrach and E. Porat. Path disruption games. In *AAMAS'10*, 2010.

[2] N. Basilico, N. Gatti, and F. Amigoni. Leader follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS'09*, 2009.

[3] S. Bohacek, J. Hespanha, and K. Obraczka. Saddle policies for secure routing in communication networks. In *Decision and Control, 2002*, 2002.

[4] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *EC'06*, 2006.

[5] P. S. Efraimidis and P. G. Spirakis. Weighted random sampling with a reservoir. *Information Processing Letters*, 97(5):181 – 185, 2006.

[6] E. Halvorson, V. Conitzer, and R. Parr. Multi-step Multi-sensor Hider-Seeker Games. In *IJCAI'09*, 2009.

[7] E. Israeli and R. K. Wood. Shortest-path network interdiction. *Networks*, 40:97–111, 2002.

[8] M. Jain, D. Korzhyk, O. Vanek, V. Conitzer, M. Pechoucek, and M. Tambe. A double oracle algorithm for zero-sum security games on graphs. In *AAMAS'11*, 2011.

[9] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, F. Ordóñez, and M. Tambe. Computing optimal randomized resource allocations for massive security games. In *AAMAS'09*, 2009.

[10] M. Mavronicolas, V. Papadopoulou, A. Philippou, and P. Spirakis. A network game with attackers and a defender. *Operation Research*, 43(2):243–251, 1995.

[11] H. B. McMahan, G. J. Gordon, and A. Blum. Planning in the presence of cost functions controlled by an adversary. In *ICML-2003*, 2003.

[12] S. Okamoto, P. Paruchuri, Y. Wang, K. Sycara, M. Srivatsa, and J. Marecki. Multiagent communication security in adversarial settings. In *IAT'11*, 2011.

[13] P. Paruchuri, J. Pearce, J. Marecki, M. Tambe, F. Ordoñez, and S. Kraus. Playing games with security: An efficient exact algorithm for Bayesian Stackelberg games. In *AAMAS'08*, 2008.

[14] B. V. Stengel and S. Zamir. Leadership with commitment to mixed strategies. Technical report, London School of Economics, 2004.

[15] J. Tsai, Z. Yin, J. Kwak, D. Kempe, C. Kiekintveld, and M. Tambe. Urban Security: Game-Theoretic Resource Allocation in Networked Physical Domains. In *AAAI'10*, 2010.

[16] A. Washburn and K. Wood. Two-person zero-sum games for network interdiction. *Operation Research*, 43(2):243–251, 1995.

[17] Z. Yin, D. Korzhyk, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. nash in security games: Interchangeability, equivalence, and uniqueness. In *AAMAS'10*, 2010.

Session 4E
Game Theory IV

Task Routing for Prediction Tasks

Haoqi Zhang[*], **Eric Horvitz**[†], **Yiling Chen**[*], and **David C. Parkes**[*]
[*]Harvard SEAS
Cambridge, MA 02138, USA
{hq, yiling, parkes}@eecs.harvard.edu
[†]Microsoft Research
Redmond, WA 98052, USA
horvitz@microsoft.com

ABSTRACT

We describe methods for routing a prediction task on a network where each participant can contribute information and route the task onwards. *Routing scoring rules* bring truthful contribution of information about the task and optimal routing of the task into a Perfect Bayesian Equilibrium under common knowledge about the competencies of agents. Relaxing the common knowledge assumption, we address the challenge of routing in situations where each agent's knowledge about other agents is limited to a local neighborhood. A family of *local routing rules* isolate in equilibrium routing decisions that depend only on this local knowledge, and are the only routing scoring rules with this property. Simulation results show that local routing rules can promote effective task routing.

Categories and Subject Descriptors

J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*

General Terms

Algorithms, Economics, Theory

Keywords

Scoring rules, task routing, social networks

1. INTRODUCTION

Organizations rely on a mix of expertise and on means for identifying and harnessing expertise for completing different kinds of tasks. The ability to leverage the expertise and interests of individuals effectively is crucial for the success of an organization. Accomplishing a task may require the expertise of multiple actors, and harnessing that expertise requires identifying who the experts are and providing proper incentives for inducing contributions.

One approach to coordinating expertise is to pool knowledge about competencies and preferences and to assign tasks in a centralized manner. Another approach is to rely on individuals distributed across an organization to select tasks themselves. Both approaches have flaws. In the former, an organization or system may not know which individuals have the required expertise. In the latter, while individuals may often be able to gauge their own expertise, they may not know which tasks best match their respective competencies.

In social networks and organizations, an individual's knowledge extends beyond their own expertise on tasks and topics to knowledge about the expertise of others. For example, members within the same research group know whom within that group can best review a paper, or best contribute to answering a research question. Members of a social network may know who among their friends can best answer a particular question, or otherwise provide valuable opinions on a topic of discussion. Even in situations where an individual cannot identify an expert who can best contribute to a task, they may know others who would likely know experts, or be able to identify subsets of individuals among whom the requisite expertise is likely to exist (e.g., people who share a particular interest).

We explore principles and methods for *task routing* that aim to harness the ability of people or automated agents to both contribute to a solution, and to route tasks to others who they believe can also effectively solve and route. Task routing provides an interesting paradigm for problem solving in which individuals become engaged with tasks based on their peers' assessments of their expertise. On the task level, effective task routing aims to take advantage of agents' knowledge about solving problems as well as agents' knowledge about other agents' abilities to contribute. Agents make routing decisions in a peer-to-peer manner, and the system rewards participating agents for their contributions. On the organizational level, task routing may provide a means for bringing tasks to individuals effectively, where agents' routing decisions take into account not only an individual's expertise on the particular task, but also their ability to contribute as a router.

Methods for automated and manual routing of tasks have been employed in online networks. For example, question-answering services such as Aardvark [10] allow a user to ask questions in natural language, which the system interprets and automatically routes to appropriate individuals in the user's social graph based on an assessment of who is best able and willing to provide an answer. Aardvark also allows for peer routing, where a user can manually route questions to others, enabling the system to reach users outside its fund of knowledge about people and their expertise.

We consider methods for routing and solving tasks with a focus on the challenge of efficiently obtaining accurate probability assessments about an uncertain event. For this task,

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

a question is passed among individuals on a network, and each participant can update the posterior probability and forward the task to a neighbor. We introduce *routing scoring rules* for incentivizing contributions. Given an assumption of common knowledge about the amount of information held by each agent on the network, we obtain truthful reporting of posterior probability assessments and optimal routing in a Perfect Bayesian Equilibrium. Even with this common knowledge assumption, we find that the equilibrium strategy on a general network requires finding a routing path through individuals that have the most information in aggregate and is NP-hard. In comparison, a myopic routing rule is optimal on simple topologies such as cliques.

A second difficulty is that common knowledge is unlikely to hold for large social networks where each agent's information about the competencies of others is limited to a local neighborhood (e.g., friends, and perhaps friends of friends). To handle such cases, we also consider task routing where knowledge about others' abilities may be limited to only those agents in an agent's local neighborhood. Unfortunately, equilibrium routing under the routing rules becomes even more computationally challenging, requiring an agent on the path to perform inference that takes into consideration the previous routing decisions of agents.

Beyond formalizing the joint routing and solution challenge, our main contribution is to introduce a family of *local routing rules* that isolate simple routing decisions in equilibrium, while still taking advantage of knowledge about the expertise of others to promote effective routing decisions. We achieve this by incentivizing agents to make routing decisions based on short, locally optimal paths that can be computed easily using shared local knowledge. In summary, *we design incentive schemes that explicitly enable equilibrium behavior for which the inference required of agents is tractable.*[1] We provide a full characterization of local routing rules, and show that they are the only routing scoring rules that induce truthful equilibria in which agents' routing decisions can be computed using only local common knowledge. Simulation results demonstrate that equilibrium routing strategies based on local routing rules lead to effective information aggregation.

1.1 Related work

Leveraging individuals' abilities to both solve and route is a key component of the winning team's strategy in the DARPA Red Balloon Challenge [13]. The task was to find large, salient helium-filled balloons placed in ten undisclosed locations across the continental United States. The winning team introduced an incentive mechanism that uses a limited budget to incentivize individuals to look for balloons and to let their friends know about the task; see also Emek et al. [8] and Douceur and Moscibroda [7] for related theoretical analysis, and work on query incentive networks [12, 2, 5] that analyze games in which players split rewards to recruit others to answer a query. The mechanism used by the winning team differs from those in our work because it aims to induce agents to broadcast the task to everyone they know regardless of their expertise or knowledge of others' expertise, whereas mechanisms in our work aim to induce agents to identify particular experts that can best contribute to the task and route it to others.

[1]This is analogous to the role of strategyproofness in simplifying strategic problems facing agents in mechanism design.

The problem of task routing is also related to the problem of decentralized search on networks in which the goal is to find a target node quickly through local routing decisions [15, 6, 16, 11, 1]. In such work, the goal is to identify a *single* target node representing a particular individual; while a single-target task differs from the task routing problem we seek to solve, the results on its solution provides theoretical and experimental support for the prospect that routing decisions with local information may have effective, global performance.

One can view routing scoring rules as an extension of market scoring rules [9] used in prediction markets. Market scoring rules provide proper incentives for individuals to improve probability estimates by contributing additional information. The major difference between task routing and a prediction market is in the 'burden' of identifying expertise: while prediction markets place the responsibility on individuals to find prediction tasks for which they have useful information, task routing incentivizes individuals to notify others with appropriate expertise who may otherwise be unaware of the task.

2. MODEL

To formalize the setting, consider a single prediction task T, for which we would like to gather an accurate probability assessment of the true state $\omega \in \Omega$. The probability assessment task can be for any state of the world that will be revealed later in time, e.g., "Will it snow next Tuesday in Boston?" or "Will the Celtics win the NBA championship this year?" We consider discrete state spaces, and assume without loss of generality a binary state space, such that $\Omega = \{Y, N\}$.

Consider a game with n players, where each player is represented by a node on the *routing graph* $G = (V, E)$. Edges in the graph may be directed or undirected, and indicate whether a particular player can route the task to another player. The task is initially assigned to a source player named player 1, with later players on a routing path numbered sequentially. The source player is either determined by the system, or the individual who originally posed the task. The source player is asked to update the probability of state Y from the prior probability p^0 to some probability p^1, and, in addition, to route the task to a neighbor. The selected neighbor is then asked to update the assessment p^1 to p^2 and route the task to a neighbor, and so on, until the game ends after a pre-specified number of rounds R that denotes when a final assessment must be made. We assume players receiving the task are provided with a list of people or agents who have participated so far, as well as information about the number of rounds that remain. *Our goal is to design incentive mechanisms that will induce each player to update probability assessments truthfully and route the task to other players that can best refine the prediction, so as to arrive at an accurate assessment after R rounds.*

We model players' knowledge about the task as follows: the true state of the world is drawn according to the probability distribution $\Pr(Y) = p^0$ and $\Pr(N) = 1 - p^0$, which is common knowledge to all players. While no player observes the true state directly, each player may receive additional information about the true state. To model this state of affairs, each player privately observes the outcome of some number of coin flips drawn according to a commonly known distribution that depends on the true state. Different players may observe different numbers of coin flips, where players observing more coin flips are *a priori* more knowledgeable.

Formally, we represent player i's signal c_i as a random bit vector of length l_i, where bit c_{ik} is a random variable over the outcome of the k-th coin flip observed by player i. We assume the value of bits of signal are *conditionally independent* given the true state, and drawn from the same distribution (known to all players) for all players and all bits, such that $\Pr(c_{ik} = H|\omega) = \Pr(c_{jm} = H|\omega)$ for all players i, j, bits k, m, and realization H (head). Each bit of signal is assumed to be *informative*, that is, $\Pr(c_{ik} = H|\omega = Y) \neq \Pr(c_{ik} = H|\omega = N)$ for all i, k. We also assume that bits of signal are *distinct*, that is, $\Pr(\omega = o|c_{ik} = H) \neq \Pr(\omega = o|c_{ik} = T)$ for all i, k, o, where H is heads and T is tails.[2] We assume the realization of each player's signal is private, but make different assumptions about the knowledge of one player about the *number* of coin flips of another player.

With conditionally independent signals, each player can properly update the posterior probability without having to know the signals of previous players or their length, as long as previous updates were done truthfully [4]. This is useful practically in that players do not have to keep track of nor communicate their signals, and can simply report an updated posterior probability that sufficiently summarizes all information collected thus far.

3. ROUTING SCORING RULES

With rational, self-interested players who have no intrinsic value (or cost) for solving or routing a particular task, effective task routing requires mechanisms that will incentivize players to both truthfully update and report posterior probabilities and to route tasks to individuals who can best refine the predictions of the tasks. In this section, we review strictly proper scoring rules and market scoring rules for incentivizing truthful reports, and introduce routing scoring rules, which also incentivize effective routing decisions.

In the forecasting literature, *strictly proper scoring rules* [14] are mechanisms that strictly incentivize a forecaster to truthfully reveal his subjective probability of an event, typically under the assumption that agents are risk neutral. The outcome of the event is assumed observable in the future, and payments are conditioned on the outcome. A well-known strictly proper scoring rule is the *quadratic scoring rule*, under which a player reporting probability q for state Y is rewarded $1 - (1 - q)^2$ when the true state is Y and $1 - q^2$ when the true state is N. Other strictly proper scoring rules include the logarithmic and spherical scoring rules, and any strictly proper scoring rule can be scaled or normalized via linear transformations to form another strictly proper scoring rule [3].

Market scoring rules [9] extend strictly proper scoring rules to settings where we wish to aggregate information across multiple people. Given a sequence of reports, player i reporting p^i is rewarded $s_i - s_{i-1}$, where s_i denotes the score of player i as computed by some strictly proper scoring rule applied to this agent's report alone. Note that since strictly proper scoring rules incentivize accurate reports, a player's reward under a market scoring rule is positive if and only if he improves the prediction.

[2]These assumptions rule out degenerate cases and can be made without loss of generality. A signal that is not informative can be removed from the signal space, and two signals that are not distinct can be treated as the same signal.

Building on market scoring rules, we introduce *routing scoring rules* to incentivize accurate predictions, along with effective routing decisions.

Definition 1. *A* **routing scoring rule** *defines a sequence of positive integers $k_1, \ldots, k_{R-1}$, which rewards players $i \in \{1, \ldots, R-1\}$ on the routing path:*

$$(1 - \alpha)s_i + \alpha s_{i+k_i} - s_{i-1}$$

where s_i is the score under an arbitrary strictly proper scoring rule, $\alpha \in (0, 1)$ is a constant, and $i + k_i \leq R$ for all players i. Player R reports but does not route and is paid $s_R - s_{R-1}$.

In a routing scoring rule, player i's payment is based on the marginal value the player provides for refining the prediction, as measured by his report and the report of the player who receives the task k_i steps after him, in comparison to the report of the player just before him. For player 1, s_0 denotes the score computed with respect to the prior p^0. Each player i can be paid for up to $R - i$ steps forward, and the final player R does not route and is paid by the market scoring rule $s_R - s_{R-1}$.

Intuitively, routing scoring rules reward players who are experts and players who are knowledgeable about the expertise of other players. We introduce here several routing scoring rules of particular interest. We first consider the *myopic routing scoring rule* (MRSR), which sets $k_i = 1$ for all players $i < R$. This routing scoring rule aims to reward a player for submitting accurate probability assessments and routing in a greedy manner to the player who can most accurately refine the probability assessment.

Lemma 1. *The total payment from the system in the routing game with MRSR is $s_R - s_0 + \alpha(s_R - s_1)$.*

The lemma follows from taking telescoping sums, and states that, for MRSR, the center needs to only pay for the difference between the final assessment and the initial assessment, since each player is only paid for the additional information they provide and their routing decision.

We can extend the MRSR to reward players' routing decisions based on the accuracy of information after $k_i = \min(k, R - i)$ more players have provided their information. The *k-step routing scoring rule* (kRSR) rewards a player based on his report, as well as the eventual consequence of his routing decision k steps into the future. Unlike MRSR, kRSR rewards players for routing to players who may not have information themselves but who are still able to route to others who do.

In particular, when player i's routing payment is based on player R's score, that is, $i + k_i = R$, for all i, we call this the *path-rewarding routing scoring rule* (PRSR). As its name suggests, this routing scoring rule seeks to focus a player's attention on the final consequence of his routing decision, as judged at the end of the solving and routing process.

The choice of routing scoring rule affects players' routing decisions in equilibrium, which in turn affects how much information is aggregated. To see the connection between a player's score and the amount of information aggregated, note that the expected score is strictly increasing in the total number of coin flips collected:

Lemma 2. *Let S' and S'' denote two possible sequences of players through the first k rounds of the routing process that* *are identical up to player $i < k$. Assume all players truthfully update posterior probabilities, and that player i knows l_j for players $i < j \leq k$ on S' and S''. Let $E^i_S[s_k]$ denote player i's ex-ante expectation of the score after player k's report in path S. $E^i_{S'}[s_k] > E^i_{S''}[s_k]$ holds if and only if $\sum_{m \in u(S')} l_m > \sum_{n \in u(S'')} l_n$, where $u(S)$ is the (unique) set of players in S.*

PROOF. (sketch) Assume without loss of generality that there are a total of n coin flips in S', and $n + m$ coin flips in S'', $m > 0$. The expected score of player k from S'' consists of two (hypothetical) parts, (a) the score he would get when giving a prediction after receiving the first n coin flips, denoted $s_{[n]}$, and (b) the difference in the score he would get by changing his prediction after receiving the next m coin flips, denoted $s_{[n+m]} - s_{[n]}$. The expectation of the first part is the same as the expected score of player k from S', and the expectation of the second part is always non-negative given any strictly proper scoring rule. □

Intuitively speaking, additional bits of information can only improve the accuracy of the prediction in expectation. Since strictly proper scoring rules reward accuracy, collecting more coin flips will lead to higher scores in expectation.

4. CASE OF COMMON KNOWLEDGE

Having introduced routing scoring rules of interest, we consider an equilibrium analysis of the associated routing game. We first consider the case where the number of coin flips l_i observed by each player i is common knowledge.[3] Note the actual signal realizations are still assumed private.

4.1 Clique topology

Let us now consider the routing game on a *clique*, where each player can route the task to any other player. From Lemma 2, and given the clique topology, an optimal routing algorithm can just route myopically and collect as many coin flips as possible at each step. This is because there is no opportunity cost for being greedy in this way, due to the clique topology. We have the following equilibrium result:

THEOREM 1. *Assume the number of coin flips of each player is common knowledge, and that players are risk neutral. Consider a routing game in which the routing graph is a clique, and let $S_{>i}$ denote the set of players who have yet to receive the task after i rounds. Under the myopic routing scoring rule, it is a Perfect Bayesian Equilibrium (PBE) for each player i to truthfully update the posterior probability, and to route the task to player $i+1 \in \text{argmax}_{m \in S_{>i}} l_m$, with the belief that all other players update the posterior probability truthfully.*

PROOF. (sketch) We show that no player wishes to deviate from the equilibrium strategy, given the belief that all other players report truthfully. Consider player i. To prove the theorem, we first show that player i should honestly update the posterior beliefs by establishing that (a) truthful reporting maximizes s_i, and that (b) for any player m who may be routed the task, truthful reporting by player i maximizes the score s_m. For (a), note that, since s_i is based on a strictly proper scoring rule, truthful reporting maximizes the expectation of s_i. For (b), note that the expected score of s_m (from the perspective of player i) is strictly greater when player i reports honestly because s_m is based on a strictly proper scoring rule. It is left to show that player i maximizes s_{i+1} by routing to the player in $S_{>i}$ with the most coin flips; this follows from Lemma 2. □

[3]By taking appropriate expectations, the analysis throughout the paper extends easily to settings where players are equally well-informed but are uncertain about the number of coin flips that other players observe.

4.2 General networks

We now turn to consider routing games on general networks, with missing edges; e.g., only managers can route tasks between teams, only professors can route questions to other professors, and only friends can route to friends.

We can state the algorithmic problem of finding the optimal route in terms of collecting coin flips:

PROBLEM 1. *Consider the routing graph $G = (V, E)$, where nodes are assigned non-negative integer weights w_i (coin flips). Given a starting node o, find a path of length at most k such that the sum of weights on the path is maximized.*

Note that a player can route to another player who have received the task before (e.g., the path need not be *simple*), but no additional information is collected in subsequent visits.

Immediately, we see that myopic routing will not always find the optimal solution to this problem, as routing to the neighbor with the most coin flips does not consider the consequence on future routing decisions and can now convey an opportunity cost.

We can show that this problem is NP-hard for variable path length k:

LEMMA 3. *Problem 1 is NP-hard.*

PROOF. Consider a reduction from the Hamiltonian Path problem. Let all nodes have weight 1, and set $k = |V|$. The solution path has total weight $|V|$ if and only if all nodes are visited within k steps, that is, a Hamiltonian Path exists. □

While the problem is NP-hard for a variable path length k, for small constant k the optimal path may be tractable to compute via exhaustive search.

Intractability is not the only difficulty faced. Even if players can compute the optimal path, we still need to find incentives that induce players to honestly report their information and to route along the optimal path. The path-rewarding routing scoring rule does just that.

THEOREM 2. *Assume the number of coin flips of each player is common knowledge, and that players are risk neutral. Let $S_{>i}$ denote the set of players who have yet to receive the task after i rounds. Let Q_i denote a solution to problem 1 for which $k = R - i$, $o = i$, and $w_m = l_m$ if $m \in S_{>i}$ and 0 otherwise. Under the path-rewarding routing scoring rule, it is a PBE for each player i to truthfully update the posterior probability, and to route the task to the next player in the path provided by Q_i, with the belief that all other players follow this strategy.*

Since PRSR rewards each agent's routing decision based on the final score, it is in each agent's interest to maximize the number of coin flips collected along the entire routing path. We can show that reporting honestly and routing this way is the only behavior that can be supported in equilibrium under PRSR:

Theorem 3. *The set of PBE identified in Theorem 2 (corresponding to possible ties in the solution to problem 1) are the only PBE of the routing game under PRSR.*

Proof. (sketch) Given any routing path, by backward induction every player should update the posterior probability truthfully because agents' scores are computed using a strictly proper scoring rule. Given that players update truthfully, by backwards induction every player i should route along the path identified by some solution Q_i because maximizing the number of coin flips collected maximizes the routing portion of each player's score (Lemma 2). □

5. LOCAL COMMON KNOWLEDGE

Although people may know one another's expertise in small organizations, the common knowledge assumption becomes unreasonable for larger organizations and social networks. Any given individual will not necessarily know everyone else, and may only have summary information about the expertise and connectivity of individuals outside of a local neighborhood.

We replace the common knowledge assumption with a requirement that individuals all attain the same minimal level of knowledge about each others' expertise within a particular size of local neighborhood, defined by the number of hops between agents. For example, all friends of a particular person are aware of his expertise, and friends of his friends may also be aware; people may know a local portion of the routing graph, e.g., individuals typically know not only their friends but also their friends' friends.

Definition 2. *A routing game satisfies the* **local common knowledge assumption within m-hops** *if, for all nodes (individuals) i, (a) l_i is common knowledge to all individuals connected to i via some path of length at most m, and (b) i knows all paths of length at most m connecting i to other individuals, and this is common knowledge.*

For example, 1-hop local common knowledge assumes all friends of a particular person know the person's level of expertise, and 2-hop local common knowledge extends this shared knowledge to his friends of friends. Note that the local common knowledge assumption within m-hops is just a minimal requirement, and does not preclude a player having more information.

Given that a player may only have m-hop local common knowledge, let's consider the problem facing such a player in deciding how to route to maximize the final prediction quality after R steps. Routing optimally may require a player to use the history of routing decisions to infer why certain people were not routed the task (but could have been), based on which to perform inference about the amount of information of different agents in the network. Furthermore, optimal routing requires a player to make inferences about the value that can be generated from the routing decisions of subsequent players beyond his locality. Not only is such reasoning complex and likely impractical, any equilibrium to induce optimal routing is likely to be fragile as it requires players to adopt priors on other players' beliefs.

An attempt to avoid such issues may suggest incentivizing players based on a m-step routing rule whenever the local common knowledge assumption holds for m-hops. The problem with this suggestion is that a player still has to consider routing decisions of players outside its locality because maximizing its payoff requires considering the routing decisions

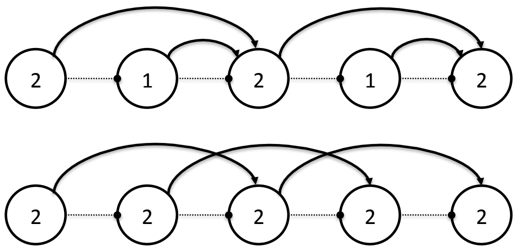


Figure 1: Illustration of the 2-1-2-1 and 2-step routing rules. Arrows depict dependencies in routing payments.

of the chain of players within its locality. For example, consider the two-step routing rule (see bottom of Figure 1). For any player, the score two steps forward will depend in part on the routing decision of the next player. But since the next player is also paid by the two-step routing rule, his routing decision will depend not only on the amount of information held by the player after him, but also that player's routing decision. Since each player has to consider the routing decision of the next player, each player has to reason about the future routing decisions of all players down the routing path, just to compute the expected score after two steps.

This motivates the family of *local routing rules*, under which players' strategies in equilibrium rely only on computations based on local information, but nevertheless take advantage of the available local common knowledge. We define the notion of a local strategy as follows:

Definition 3. *A player i in a routing game adopts a* **m-local strategy** *if its routing decision depends only on m-hop local common knowledge and is invariant to any beliefs the player might have about players outside of its own locality.*

Let us first consider the following local routing rule, designed to be useful with 2-hop local common knowledge:

Definition 4. *The* **2-1-2-1 routing rule** *is a routing scoring rule which sets $k_i = 2$ if i is odd and $i < R - 1$, and $k_i = 1$ otherwise.*

The 2-1-2-1 routing rule incentivizes players to compute locally optimal paths of length two (see top of Figure 1), which can be computed with local common knowledge, and so inference is not required. As even players are paid based on the myopic routing scoring rule, they will route to the available player with the most number of coin flips. Since each odd player knows the number of coin flips that can be collected from the next even player and also from the odd player that is then routed the task, he can compute the best local path without regard to routing decisions beyond his locality. Note that players still need to take into account which other players have already participated, but no other inference based on history is necessary.

Expanding on the idea, we construct a class of routing scoring rules (e.g., MRSR, 2-1-2-1, 3-2-1-3-2-1, ...) that incentivize players to compute *locally optimal paths* for m-hop local common knowledge:

Definition 5. *The* **m-hop routing rule** *is a routing scoring rule which sets $k_i = \min[m - (i-1) \bmod m, R - i]$.*

We can characterize the equilibrium behavior as follows:

THEOREM 4. *Assume that players are risk neutral and m-hop local common knowledge holds. Let $S_{>i}$ denote the set of players who have yet to receive the task after i rounds. Let Q_i denote a solution to problem 1 for which $k = \min[m-(i-1) \bmod m, R-i]$, $o = i$, and $w_j = l_j$ if $j \in S_{>i}$ and 0 otherwise. Under the m-hop routing rule, it is a PBE for each player i to truthfully update the posterior probability, and to route the task to the next player in the path provided by Q_i, with the belief that all other players follow this strategy.*

PROOF. (sketch) Using similar arguments as the proof sketch for Theorem 1, we can show that players should truthfully update the posterior probability. To show player i should route based on Q_i, we first note that Q_i is computable given m-hop local common knowledge. Since Q_i maximizes the number of coin flips collected in the next k steps, Lemma 2 proves the point, and the theorem. $\square$

The main idea behind the m-hop routing rule is that each player can compute his best routing action with respect to the decisions in his locality and without regard to routing decisions beyond his locality. This property can be satisfied by other local routing rules as well. For example, when $m = 3$, the 3-1-1-3-1-1 routing rule is one in which the first of three players in sequence is paid by the score three steps forward, but in which the next two players are each paid myopically. Note that here the first player can still compute its optimal routing decision using only local common knowledge, by computing the routing decisions of others in its locality via backwards induction. We can thus characterize the entire family of local routing rules:

DEFINITION 6. *Given m-hop local common knowledge, the family of* **local routing rules** *contains routing scoring rules $k_1, \ldots, k_{R-1}$ that satisfies* **local reasoning**, *that is, $k_{i+j} + j \leq m$ for all i and $0 \leq j < k_i$.*

The local reasoning condition ensures that local routing rules can only reward players whose routing decisions may affect the payoff of an earlier player based on the routing decisions of future players that are within m hops of that earlier player. In other words, it considers the set of routing scoring rules for which the payment to any player should only depend on the local information that player is guaranteed to hold. For example, the 2-1-2-1 routing rule satisfies local reasoning for $m = 2$ because for an odd i, $k_i \leq 2 \leq m$ and $k_{i+1} + 1 = 2 \leq m$, and for an even i, $k_i = 1 \leq m$. However, the two-step routing scoring rule violates local reasoning, because for all $i < R-2$, $k_{i+1} + 1 = 3 > m$. Note that the m-hop routing rule satisfies local reasoning, since k_i is set such that $k_{i+j} + j = m$ for all appropriate i and j in the above definition.

We argue that using a local routing rule is necessary and sufficient for the existence of an equilibrium in which agents follow m-local, truthful strategies. We first show sufficiency:

THEOREM 5. *Assume that risk neutrality and m-hop local common knowledge holds. For any node i and possible path $n_{i+1}, \ldots, n_{i+k_i}$ from i, let the weights w_j on node j be l_j if j has yet to be visited up until then, and 0 otherwise. For any local routing rule, consider the following dynamic program:*

$$V(n_{j+1}, \ldots, n_{j+k_j} | n_1, \ldots, n_j) = \max_{j+1,\ldots,j+k_{j+1}} [\sum_{b=1}^{k_{j+1}} w_{j+b} + V(n_{j+k_{j+1}+1}, \ldots, n_{j+k_j} | n_1, \ldots, n_{j+k_{j+1}})]$$

$$V(\emptyset | n_1, \ldots, n_{j+k_j}) = 0 \quad \forall n_1, \ldots, n_{j+k_j}$$

Let $n^*_{i+1}, \ldots, n^*_{i+k_i} = \operatorname{argmax} V(n_{i+1}, \ldots, n_{i+k_i} | n_1, \ldots, n_i)$ denote a solution of the dynamic program. It is a PBE for each player i to truthfully update posterior probabilities and to route the task to n^*_{i+1}, with the belief that all other agents follow this strategy.

PROOF. (sketch) To prove the theorem, we first note that all players would truthfully update the posterior probability along the path as we had previously argued, as doing so maximizes the scores computed, based on its assessment and based on the assessments collected from those routed the task via the routing payment. Second, as the variables and parameters of the dynamic program are only the nodes in paths of length at most k_i from i, and by the local reasoning assumption $k_i \leq m$, players follow m-local strategies in which the information that each player i needs to compute the dynamic program is within m hops and thus known to player i. Finally, given the routing decisions of others down the path, the number of coin flips collected is by definition maximized by the routing decisions along the computed path. Applying Lemma 2 proves the point, and the theorem. $\square$

THEOREM 6. *For any local routing rule, the set of PBE identified in Theorem 5 (corresponding to possible ties in the optimal solution to the dynamic program) are the only PBE of the routing game under that local routing rule.*

THEOREM 7. *The only routing scoring rules that induce for every routing game a truthful PBE (where players honestly update probability assessments) in m-local strategies are local routing rules.*

PROOF. (sketch) Assume for sake of contradiction that there exists a routing scoring rule that induces a truthful PBE for all routing games in m-local strategies but is not a local routing rule. Since this routing scoring rule violates local reasoning, there must be some i in the sequence for which there exists some j such that $k_{i+j} + j > m$, $0 \leq j < k_i$. Consider the first such i and j.

First consider the case where $j = 0$. We construct a graph with two paths (top and bottom), as is shown in Figure 2:

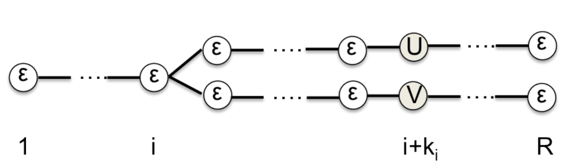


Figure 2: Routing game construction for $j = 0$ case.

Based on the construction, consider two routing games G and G', where in game G the coin flips held by U and V are 1.5ϵ and 1.6ϵ respectively and in game G' the coin flips at U and V are reversed. Due to the violation of local reasoning at i for $j = 0$, by construction U and V are more than m hops from player i. In a PBE with m-local strategies, it is thus necessary for the routing decisions of player i to be independent of the number of coin flips held by players at U and V, that is, for the routing decision to be the same for these two games G and G'.

We show that player i's best response to the equilibrium strategies of the other agents depends on G or G'. For both games, using backwards induction, all players strictly prefer to route the task forward (to the right) instead of backwards at any given point in time and for any lookahead depth as induced by their routing payment, because its expected payment is based on the number of coin flips collected and one can always collect more coin flips in the forward direction (because for any player, going backwards would necessitate visiting a node that's been visited before and thus has no new coin flips to share). Since in game G player i would collect more coin flips by routing up due to the higher value at U over V and the reverse is true in game G', player i's best response would be different, which contradicts our assumption.

Now consider the case where $j > 0$. We construct a graph with three paths (top, middle, and bottom), as is shown in Figure 3:

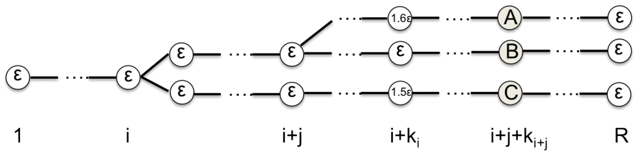


Figure 3: Routing game construction for $j > 0$ case.

Based on the construction, consider two routing games G'' and G''', where in game G'' the coin flips held by A, B, and C are ϵ, ϵ, and ϵ respectively, and in game G''' are ϵ, 1.7ϵ, and 1.7ϵ, respectively. Due to the violation of local reasoning, by construction A, B, and C are more than m hops from player i. In a PBE with m-local strategies, it is thus necessary for the routing decisions of player i to be independent of the number of coin flips held by players at A, B, and C, that is, for the routing decision to be the same for G'' and G'''.

We show that player i's best response to the equilibrium strategies of the other agents depends on G'' or G'''. We first consider game G''. Using backwards induction, note that each player must strictly prefer to route the task forward (to the right) instead of backwards at any given point in time, regardless of the lookahead induced by their routing payment, because its expected payment is based on the number of coin flips collected and, as before, one can always collect more coin flips in the forward direction (as going backwards necessitates visiting a node that's been visited before). In this case, the top player at $i + j$ would route up because the $i + k_i$-th player will have more coin flips (1.6ϵ) and is within the scope of the routing payment. Given this, it is strictly better for player i to route up instead of down, given knowledge of the values at A and B.

Consider now game G'''. By backwards induction, each player strictly prefers to route forward because doing so guarantees the largest payment along the way for any lookahead. The top player at $i + j$ will route along the middle path in equilibrium because he would receive $\epsilon + 1.7\epsilon$ from coin flips at the middle path of $i + k_i$ and $i + j + k_{i+j}$ vs. the 1.6ϵ $+\epsilon$ along the top path. In this case, player i would rather route down instead of up because it would collect 0.5ϵ more coin flips due to the 1.5ϵ at $i + k_i$ on the bottom path. However, since player i's best response routing decision should be the same for game G'' and G''', we have a contradiction. $\square$

		$d = 4$			$d = 10$		
β	Dist.	MRSR	m=2	m=3	MRSR	m=2	m=3
.03	U	69	71	72	83	84	85
0.1	U	71	72	75	85	86	87
1.0	U	76	78	80	89	89	90
.03	S	80	87	104	150	183	227
0.1	S	88	109	146	181	226	259
1.0	S	120	155	183	227	258	278

Table 1: Comparison of routing performance after 10 steps on connected Watts-Strogatz graphs based on uniform (U) and skewed (S) coin flip distributions with fixed mean (5.5).

6. SIMULATION RESULTS

The equilibrium strategies induced by local routing rules can be considered to provide a heuristic algorithm for computing an optimal route over a network. We now demonstrate via simulations that routing decisions based on local rules can effectively aggregate information as a task is routed through the network.

We consider connected random graphs with 100 nodes and average degree $d \in \{4, 10\}$, generated using the Watts-Strogatz model [17]. By varying the re-wiring probability β, the model allows us to generate graphs that interpolate between a regular lattice ($\beta = 0$) and a $G(n, p)$ random graph ($\beta = 1$), with small-world networks emerging at intermediate values of β. We associate each node with a number of coin flips, which is drawn independently either discretely from U[1,10], or from a skewed distribution where the value is 1 with probability 0.9 and 46 with probability 0.1. Note that the distributions have equal mean (5.5), but that the skewed distribution more closely resembles a setting where there are few experts. For graphs generated in this manner, we simulate player strategies under local routing rules (MRSR, and m-hop with $m = 2$, $m = 3$) by computing local paths in the manner noted in Theorem 4, where revisited nodes are treated as having no value. As a baseline, we consider a random routing rule that routes to a random neighbor, and whenever possible, to a random neighbor who has yet to be assigned the task. Note that the expected performance of the baseline is bounded by 5.5 coin flips per round, as we would expect from randomly picking unvisited nodes in the graph.

Table 1 shows the average number of coin flips collected after 10 steps by players following local routing rules on graphs with varying β, average degree, and coin flip distribution over 100 trials (standard errors are small and hence not reported). We see that routing rules are particularly effective in cases where there are few experts (S), and when the graph has a sufficiently high connectivity (higher d and β) that there exist paths through which experts can be routed the task. But even in cases with uniformly distributed coin flips (U) and low average degree ($d = 4$), local routing rules collect significantly more coin flips than the upper bound of 55 we would expect from randomly choosing nodes, and that despite connectivity constraints the paths include many high valued nodes (recall the max per node is 10).

The difference in routing performance among local routing rules is rather small for uniformly distributed values, but is more significant when the distribution is skewed. In

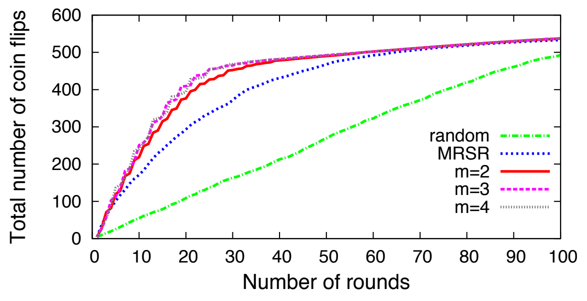


Figure 4: Comparison of routing performance among local routing rules for graphs with $\beta = 0.1$, $d = 10$, and skewed coin flip distributions. Values are averaged over 100 trials.

this case, effective routing may require finding short paths to experts who are not neighbors. That said, this difference shrinks for graphs with higher degree, as high-value nodes become more easily reachable (recall that as graphs approach cliques, myopic is optimal). Figure 4 shows the average number of coin flips collected by local routing rules as we progress through the routing game on graphs with $\beta = 0.1$, $d = 10$, and skewed coin flip distributions. We see that for $m \geq 2$ the performance under the routing rules are essentially the same, suggesting that we can sometimes achieve near-optimal performance globally with just two-hop local common knowledge. Note that for all local routing rules the rate of information aggregation eventually slows down, which denotes the point at which virtually all experts have been routed the task.

7. CONCLUSION

We consider the opportunity for incentivizing the joint refinement and routing of tasks among agents within a network, focusing on prediction tasks. We introduce and study routing scoring rules which, in equilibrium, support agents truthfully contributing information, and routing tasks based on simple computations that nevertheless lead to effective information aggregation. Future work on task routing for prediction tasks includes efforts to integrate additional information structures and study routing performance under specialized network topologies, consideration of intrinsic values for solving or routing, and introduction of communication or sensing mechanisms coupled with means of tracking costs for acquiring additional bits of signal. There are multiple opportunities to address task-level issues, and also organizational issues related to distributing streams of tasks in a manner that takes into account people's solving and routing abilities over a spectrum of tasks, as well as participants' changing levels of attention, motivation, and availability, and the corresponding need for balancing the load across participants. We envision numerous potential applications of methods for jointly solving and routing tasks and foresee an ongoing need to strike insightful balances between principled procedures and designs that rise from intuitions about practical implementations.

Acknowledgments

The problem was initially formulated at Microsoft Research during an internship there by Haoqi Zhang. Haoqi Zhang is funded by a NSF Graduate Research Fellowship. The research was also sponsored in part by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053 and by National Science Foundation under Grant Number CCF-0953516. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

8. REFERENCES

[1] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman. Search in power-law networks. *Phys. Rev. E*, 64:046135, Sep 2001.

[2] E. Arcaute, A. Kirsch, R. Kumar, D. Liben-Nowell, and S. Vassilvitskii. On threshold behavior in query incentive networks. In *Proc. EC '07*, 2007.

[3] J. E. Bickel. Some comparisons among quadratic, spherical, and logarithmic scoring rules. *Decision Analysis*, 4:49–65, June 2007.

[4] Y. Chen, D. M. Reeves, D. M. Pennock, R. D. Hanson, L. Fortnow, and R. Gonen. Bluffing and strategic reticence in prediction markets. In *WINE*, 2007.

[5] D. Dikshit and N. Yadati. Truthful and quality conscious query incentive networks. In *WINE*, 2009.

[6] P. S. Dodds, R. Muhamad, and D. J. Watts. An experimental study of search in global social networks. *Science*, 301(5634):827–829, 2003.

[7] J. Douceur and T. Moscibroda. Lottery trees: Motivational deployment of networked systems. In *SIGCOMM '07*, 2007.

[8] Y. Emek, R. Karidi, M. Tennenholtz, and A. Zohar. Mechanisms for multi-level marketing. In *Proc. EC '11*, pages 209–218, New York, NY, USA, 2011. ACM.

[9] R. Hanson. Logarithmic market scoring rules for modular combinatorial information aggregation. *Journal of Prediction Markets*, 1(1):3–15, February 2007.

[10] D. Horowitz and S. D. Kamvar. The anatomy of a large-scale social search engine. In *Proc. WWW '10*, pages 431–440, New York, NY, USA, 2010. ACM.

[11] J. Kleinberg. Complex networks and decentralized search algorithms. In *Proc. International Congress of Mathematicians*, 2006.

[12] J. Kleinberg and P. Raghavan. Query incentive networks. In *FOCS '05*, 2005.

[13] G. Pickard, W. Pan, I. Rahwan, M. Cebrian, R. Crane, A. Madan, and A. Pentland. Time-critical social mobilization. *Science*, 334(6055):509–512, 2011.

[14] R. Selten. Axiomatic characterization of the quadratic scoring rule. *Experimental Economics*, 1(1):43–61, June 1998.

[15] J. Travers and S. Milgram. An experimental study of the small world problem. *Sociometry*, 32:425–443, 1969.

[16] D. J. Watts, P. S. Dodds, and M. E. J. Newman. Identity and search in social networks. *Science*, 296(5571):1302–1305, 2002.

[17] D. J. Watts and S. H. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393(6684):440–442, June 1998.

Mastering multi-player games

Yossi Azar[*]
Blavatnik School of Computer Science
Tel-Aviv University
Tel-Aviv, Israel
azar@post.tau.ac.il

Uriel Feige[†]
Department of Computer Science and Applied Mathematics
the Weizmann Institute
Rehovot 76100, Israel
uriel.feige@weizmann.ac.il

Michal Feldman[‡]
Hebrew University
and Harvard University
Jerusalem, 91905, Israel
mfeldman@huji.ac.il

Moshe Tennenholtz
Microsoft Research and Technion
13 Shenkar street
Herzliyah, Israel
moshet@microsoft.com

ABSTRACT

We consider multi-player games, and the guarantees that a master player that plays on behalf of a set of players can offer them, without making any assumptions on the rationality of the other players. Our model consists of an $(n+1)$-player game, with m strategies per player, in which a *master* player M forms a coalition with nontransferable utilities among n players, and the remaining player is called the *independent* player. Existentially, it is shown that every game admits a *product-minimax-safe* strategy for M — a strategy that guarantees for every player in M's coalition an expected value of at least her *product minimax value* (which is at least as high as her minimax value and is often higher). Algorithmically, for any given vector of values for the players, one can decide in polytime whether it can be ensured by M, and if so, compute a mixed strategy that guarantees it. In symmetric games, a product minimax strategy for M can be computed efficiently, even without being given the safety vector. We also consider the performance guarantees that M can offer his players in repeated settings. Our main result here is the extension of the oblivious setting of Feldman, Kalai and Tennenholtz [ICS 2010], showing that in every symmetric game, a master player who never observes a single payoff can guarantee for each of its players a *similar* performance to that of the independent player, even if the latter gets to choose the payoff matrix after the fact.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*; J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*

General Terms

Algorithms, Economics, Theory

Keywords

Game Theory, minmax, repeated games

1. INTRODUCTION

In many situations, a *master* is playing on behalf of multiple players, and wishes to offer them some guarantees. For example, a representative plaintiff in a class action represents a group of complainants, and wishes to guarantee the individual complainants a certain level of compensation. In many of these settings, the payoffs of the players under the master's control are nontransferable, due to the nature of the good or some legal or ethical concerns. Presumable, every individual player has some safety-level payoff, which is the payoff she can guarantee for herself, when playing independently. An individual player will typically not delegate her action to a third party unless he can guarantee for her a value of at least the safety level she can guarantee for herself. The focus of this paper is the performance guarantees that a master can provide to the members of the coalition he forms. This involves two aspects. One is *existential*, studying which guarantees are possible. The other is *algorithmic*, presenting efficient algorithms that find strategies for M.

As is well known [1], if the master can play on behalf of all the players in the game, he can offer them a correlated equilibrium. However, in this manuscript the master plays only on behalf of some of the players, and not all. Hence he forms a coalition among the players that he controls. The strength of coalitions has been studied in the past [2, 8], but mostly

[*]Supported in part by the Israel Science Foundation (grant No. 1404/10).

[†]This research was conducted while at Microsoft Research at Herzliyah.

[‡]This research was partially conducted while at Microsoft Research at Herzliyah. This work was partially supported by the Israel Science Foundation (grant number 1219/09), by the Leon Recanati Fund of the Jerusalem School of Business Administration, and by the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement number 274919.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

in the context of incentives they may have to deviate from a Nash equilibrium. In this manuscript we assume nothing about what other players play, and ask for guarantees to the coalition players regardless of what non-coalition players do. As we assume nothing about the strategies played by non-coalition players, we group them into one player.

To study this problem, we use the following model. Let G be an arbitrary $(n+1)$-player game with m strategies per player. Consider a situation in which a *master* player M controls players $P_1, \dots P_n$, and the remaining player P_0 is called the *independent* player I. This induces a new two player game between the master player and the independent player. This new game is referred to as the *Master Game* of G, and is denoted G_2. The payoffs for the independent player in G_2 are the same as in G. For the master player, there is a vector of payoffs, one for each player P_i. The goal of M is to provide performance guarantees to the players under his control. Crucially, throughout the paper, the notion of performance *guarantee* refers to the *expected* value that can be guaranteed.

What types of guarantees can a player expect to get? There are various safety notions that correspond to different guarantees a player can guarantee for herself when playing as an individual. Clearly, the master can guarantee each player the minimum possible payoff in the payoff matrix for that player. But in general, can the master offer better guarantees? Two natural candidates that do not work are the following:

1. For every $i \in [n]$, guarantee player P_i the minimum payoff for P_i in any Nash equilibrium. This does not work because I is not forced to play an equilibrium strategy. We wish our guarantees to hold even if I plays arbitrarily.

2. For an $(n+1)$ player game G and a choice x of $n-1$ pure strategies, let G_x^i be the marginal game that remains between player P_i and P_0 when the remaining players play the strategies as in x, and the payoff for P_0 is the negative of the payoff for P_i. The suggested guarantee is to guarantee every player P_i the minimum over x of the value of the games G_x^i. To see that this cannot be guaranteed, consider for example a three player game in which P_1 and P_2 are playing matching pennies with each other with payoffs ± 1, and P_0 only observes (is irrelevant to the payoffs). Hence given what P_2 plays (this is x for P_1), P_1 can get a payoff of 1 ("against P_0") and vice versa, but clearly there is no strategy for M controlling P_1 and P_2 that ensures expected payoff of at least 1 for both players simultaneously.

Additional natural safety notions for a player P_i are the *minimax* and *maximin* values. The minimax value is the expected value that P_i can guarantee if the other players announce a (possibly correlated) strategy first and P_i responds, and the *maximin* value is the expected value that P_i can guarantee if she announces a strategy first and the other players respond. It is also interesting to consider the *product minimax* and *product maximin* values, which are the respective versions of minimax and maximin, where the other players are restricted to play a product mixed strategy.

One may observe that the definitions of maximin value and product maximin value are in fact identical. Once a player announces her mixed strategy, the most harmful response is a pure strategy on behalf of each of the remaining players, and hence is captured by a product strategy. The minimax value and maximin value are also identical, by the minimax theorem for two player games (here the player under consideration is one player and a coalition of all remaining players serves as the other player). Hence the maximin, minimax and product maximin notions are equivalent to each other. However, the product minimax value might be different; it is at least as high, and often higher.

In simultaneous play, a player may guarantee for herself the minimax value (which equals the maximin value and the product maximin value). However, a player cannot guarantee the product minimax value, at least not without making assumptions on the rationality of other players. We ask whether a master who plays on behalf of n players can guarantee for each one of them her product minimax value.

Our existential result asserts that for every game G, there exists a mixed strategy for M in G_2 that guarantees for every player P_i under his control her product minimax value. Such a strategy for M is called a *product-minimax-safe* strategy. Moreover, we show that given any vector $\bar{v} = (v_1, \dots, v_n)$, one can find in polynomial time whether there exists a mixed strategy for M that guarantees for every player P_i a value of v_i, and if so, compute it in polynomial time. Consequently, given a vector of the players' product minimax values, a product-minimax-safe strategy can be computed in polynomial time. One should, however, not misinterpret the last result to suggest that the master can always compute a product-minimax-safe strategy in polynomial time. Indeed, it is shown in [5] that computing the product minimax value of a player is NP-hard, even for symmetric games. This makes it difficult for M to deduce what are the product minimax values that he needs to attain for his players. It remains open whether a product minimax safe strategy can be found without the safety vector being given.

Much of the intuition of what can and cannot be achieved is given through the analysis of *symmetric games.* As a special case of our existential result, it follows that in every symmetric game, M can guarantee each one of his players an expected payoff that equals at least that of I. Moreover, in the symmetric case, the master can also compute in polynomial time (via a linear program) a strategy that guarantees each one of his players at least her product minimax value. Interestingly, there exist symmetric games in which M can guarantee each one of his players an expected payoff that is strictly greater than the payoff of I. The last result is particularly significant in competitive settings, where players care less about their absolute payoff, rather they wish to perform well relative to others. For example, when a group of potential employees compete over a limited set of positions, they are mostly concerned with their ranking within the group.

In addition to the *vector version* (where the payoff of M is a vector of his players' payoffs), we consider the *minimum version*, where the payoff for M is the minimum of the payoffs to all his players. This version models situations that exhibit the "weakest-link" characteristic — where the weakest player determines the outcome for all involved parties. A classical example, given by [9], is the level of flood protection of an island, which is determined by the lowest dike along the coastline. Other prominent examples include airport security, the spread of infectious diseases, and more. Unfortunately, the guarantees that can be offered in the vector version do not extend to the minimum version. Indeed, there exist symmetric games in which for every mixed strategy of M, there exists a player P_i whose expected payoff is strictly lower than I's payoff.

In the second part of the paper we extend the model to repeated settings under various information structures. In all settings, a two-player repeated game between M and I is considered, where the payoff for M is a vector of his players' payoffs, and I is assumed to have full information of the payoff matrix and the history of play. The various settings differ by the information available to M, and the nature of the desired safety vector depends on the specific setting. We say that M approaches a safety vector $\bar{v} = (v_1, \ldots, v_n)$ if regardless of the strategy of I, the expected sum of payoffs over T rounds, for every player P_i controlled by M, is $Tv_i - o(T)$.

We first consider a full information setting. In a 2-player game, where the payoff of player 1 is a vector, Blackwell's approachability theorem [4] implies that a vector is *vector-minimax achievable* (in a one-shot game) if and only if it is also approachable (in a repeated setting). This implication can be viewed as a generalization of the minimax theorem for vector payoffs. In our setting, the payoff of M is the vector of its players' payoffs; thus it is essentially a 2-players game between M and I, with M having a vector payoff. In the one-shot game, the payoff vectors that are *safe* for M are precisely those that are *vector-maximin achievable*. For example, by our existential result for the one-shot game, the class of vectors that consist of respective product-minimax values of the players are vector-maximin achievable in the one-shot game. Vectors that are minimax achievable in the one shot game are not necessarily safe for M in the one shot game. Nevertheless, based on the implication of Blackwell's theorem stated above, these vectors are approachable in the repeated settings. Thus, the set of safe vectors for M in the repeated setting is larger than the set of safe vectors in the one-shot game.

We also consider an *oblivious* version, in which payoffs are never observed by M over the course of the game, extending the setting of [7] from two players to multiple players. This version models situations where players are engaged in games but the payoffs are determined only in later stages. This is the situtaion, for example, in cases where the actual behavior of players is evaluated by other parties only in retrospect or when the actual payoffs are determined based on an event whose outcome is unknown in the time actions are taken. A motivating example would be a reporter who writes a daily column about political issues. In some countries reporters may be recruited after the elections to represent government officials. Needless to say that the content of the daily columns by the different reporters may affect this selection. However, only after the government is selected the reporters will be able to determine their payoffs for their expressed opinions.

If players have similar capabilities and preferences, then these games are also symmetric. In the latter case an attractive objective for a player is to guarantee a payoff which is close to the one of his competitor despite the a-prior lack of information; indeed, for the case of two players this problem has been addressed by [7]. More generally, however, we ask whether a group of players can guarantee each one of them a payoff close to the one of an independent player, even if that player is all capable and able to predict the actual payoff function.

To model such situations, we consider a repeated game between M and I, where the game is first played T times, and after all T repetitions are completed I gets to choose the payoff matrix. The notion of safety for M that we use here is that as T tends to infinity, the regret is $o(T)$. We show that for every symmetric game G, M is safe in the oblivious repeated version. Moreover, a safe strategy for M can be found in time polynomial in m^n and T.

Related work.

Our work relates to the body of work on mediators in AI (see e.g. [10, 12]), which extends on classical literature on non-transferable utility coalition games, originated in [3], which relies on the fundamental work introduced in [2] (leading to the concepts of β-core and c-acceptable strategies, respectively). In particular, given a non-cooperative game, a c-acceptable strategy can be viewed as a recommended strategy profile to the group of all agents, augmented with a punishing correlated strategy for each strict subset of the agents. The idea is that the recommended strategy will be accepted if there is no subset of the players that can all gain with respect to this strategy payoffs, using a correlated strategy without side-payments when playing against the punishing correlated strategy of the remaining agents. A subset of deviating agents can be viewed as a master player, and the question is whether this master player can guarantee a vector of payoffs that exceeds the corresponding vector of payoffs of the recommended strategy profile. Some of our results complement this literature, by showing that if the recommended strategy payoffs are dominated by the product minimax outcomes associated with a particular master-player then it can not be accepted. We are not aware of a similar result.

2. SOME SAFETY NOTIONS

Given a multiplayer game G with n players, let us review some safety notions that a player may wish to achieve, and in some cases may indeed achieve. We use the following notation. S_i denotes a mixed strategy for player P_i. S^{-i} denotes a mixed strategy for all players except for P_i, when we do not assume that this is a product strategy. That is, all remaining players may collaborate to agree on a joint correlated mixed strategy. When the players each individually plays a mixed strategy, then $S^{-i\times}$ denotes the resulting product strategy (excluding player P_i). We use $v_i(S_i, S^{-i})$ to denote the expected payoff of player P_i when it plays the mixed strategy S_i and other players play the mixed strategy S^{-i}.

We consider here four different versions of safety notions, three of which are equivalent to each other.

- Minimax value (a best response notion). The other players announce their most harmful mixed strategy (min), and in response P_i chooses his best strategy (max). $\text{minimax}_i(G) = \max_{S_i} \min_{S^{-i}} v_i(S_i, S^{-i})$. Without loss of generality, S_i can be a pure strategy here.

- Maximin value (a best bid notion). P_i announces a mixed strategy that is deemed best (max), and in response the other players choose their most harmful strategy (min). $\text{maximin}_i(G) =$

$\min_{S^{-i}} \max_{S_i} v_i(S_i, S^{-i})$. Without loss of generality, S^{-i} can be a pure strategy here.

- Product minimax value. The other players announce their most harmful product mixed strategy, and in response P_i chooses his best strategy. $\text{minimax}_i^{\times}(G) = \max_{S_i} \min_{S^{-i\times}} v_i(S_i, S^{-i\times})$. Without loss of generality, S_i can be a pure strategy here.

- Product maximin value. P_i announces a mixed strategy that is deemed best, and in response the other players choose their most harmful product strategy. $\text{maximin}_i^{\times}(G) = \min_{S^{-i\times}} \max_{S_i} v_i(S_i, S^{-i\times})$. Without loss of generality, S^{-i} can be a pure strategy here.

As mentioned in the Introduction, the minimax, maximin and product maximin values are identical, whereas the minimax value is at least as high and may be higher. The superiority of the product minimax value is demonstrated in the following example.

Example. Suppose that P_1 has four pure strategies whereas P_2 and P_3 each has two pure strategies, and hence four combinations of pairs of pure strategies. If P_1 plays strategy 1 against combination 1 its payoff is 2. For $j > 1$, if P_1 plays strategy j against combination j its payoff is 1. In all other cases the payoff of P_1 is 0. The minimax value for P_1 in this game is 2/7. The unique S^{-1} that limits P_1 to this value is the mixed strategy in which the other players play combination 1 with probability 1/7, and each other combination with probability 2/7. But this S^{-1} cannot be represented as a product strategy, and hence the product minimax value is strictly larger than 2/7.

In simultaneous play, a player may guarantee for herself the minimax value, but not the product minimax value. In this context, it is interesting to note that in every (mixed) Nash equilibrium for the multiplayer game G, every player gets at least his product minimax value. A player not getting this value (in expectation) necessarily can gain by deviating from his current mixed strategy, contradicting the assumptions of a Nash equilibrium.

We note also that there are multiplayer games for which in every Nash equilibrium, all players get expected payoff strictly larger than their respective product minimax values. For example, consider an n-player game in which each player has two strategies, and the payoffs to all players is the number of players playing strategy 1. This game has a unique Nash equilibrium, with value n to all players, but the product minimax value is only 1.

2.1 Computational complexity of safety notions

The minimax (and hence maximin and product maximin) value for a player P_i in a multiplayer game can be computed in polynomial time (time polynomial in the game matrix). This is done by viewing the setting as a 0-sum two player game, where P_i is one player, and a coalition of all players is the other player.

When the number of players is three or more, computing the product minimax value of a game is NP-hard. See [5], where the vector of product minimax values is called the *threat point*. This also holds when the 3-player game is symmetric.

3. SAFETY GUARANTEES IN ONE-SHOT GAMES

We consider multi-player games, and the guarantees that a master player that plays on behalf of a set of players can offer the players. Let G be an arbitrary $(n+1)$-player game with m strategies per player. Consider a situation in which a *master* player M controls players $P_1, \dots P_n$, and the remaining player P_0 is called the *independent* player I. Having only one independent player can be assumed without loss of generality: as we assume nothing about the strategies played by non-coalition players, we may as well view them as one player. (See Section 3.3 for a brief discussion on this point.) This induces a new two player game G_2 between the master player and the independent player. The payoffs for the independent player in G_2 are the same as in G. For the master player, there is a vector of payoffs, one for each player P_i. We use $[n]$ to denote the set $\{1, \dots, n\}$. We use the notion of *guarantee* to refer to guarantees on the *expected* payoff.

3.1 Algorithmic results

The $\bar{v}$-safe problem is defined as follows.

$\bar{v}$-safe problem. Given a multi-player game G and a vector of values $\bar{v} = \{v_1, \dots, v_n\}$, is there a mixed strategy for M that guarantees for every player P_i an expected payoff of at least v_i, and if so, present such a mixed strategy.

A key concept that we shall use here is the following.

Selection version. Given a multi-player game G we introduce a *selection version* G'. In G' players $P_1, \dots, P_n$ have the same set of strategies as in G. The set of strategies for I includes two coordinates, one being the set of strategies for P_0 and the other a choice of an index in $[n]$. Given the strategies played by the players, player P_i (where i is the value of the second coordinate chosen by I) gets the same payoff as in G, the payoff for I is the negative of the payoff for P_i, and all other players get payoff 0. The master version G'_2 of G' (where M controls $P_1, \dots, P_n$) is a 0-sum game, if the payoff for M is taken to be as the sum of payoffs of its players. This implies that linear programming can be used in order to find optimal strategies in G'_2 in time polynomial in the size of the normal form representation of G'_2, namely, polynomial in $(n+1)m^{n+1}$.

The concept of a selection version is used to show that the $\bar{v}$-safe problem if polynomial.

THEOREM 1. *There is a polynomial time algorithm for the $\bar{v}$-safe problem.*

PROOF. Given a game G and a vector $\bar{v}$, scale the payoff for each player P_i (for $i \in [n]$) by subtracting v_i, so that the $\bar{v}$-safe problem now asks for nonnegative expected payoff for every player controlled by M in G_2. We claim that this is equivalent to finding a strategy for M with nonnegative expected payoff in the corresponding selection game G'_2. If such a strategy exists for G_2, then it can be used in G'_2 and I is helpless in choosing $i \in [n]$. If such a strategy exists in G'_2, then the same strategy can be used in G_2, and every player controlled by M has expected nonnegative payoff, as otherwise there would be a strategy for I in G'_2 giving I expected positive payoff. Now polynomiality of G_2 follows from polynomiality of G'_2. $\square$

3.2 Existential results

The product minimax value of a player is always at least as high as her minimax value (and hence also the maximin and product maximin values), and is often higher (see Section 2). While a player cannot guarantee this value for herself when playing independently, we show that a master who plays on behalf of a set of players can guarantee each one of them her product minimax value.

Definition 2. *We say that a mixed strategy for M is* product-minimax-safe *for G_2 if for each player P_i controlled by M it guarantees an expected payoff of at least her product minimax value. We say that G_2 is product-minimax-safe for M if M has a product-minimax-safe mixed strategy for G_2.*

Theorem 3. *For every multi-player game G, the corresponding game G_2 is product-minimax-safe for M. Moreover, given the players' product minimax values, a product-minimax-safe strategy for M can be found in time polynomial in m^n.*

Note that the algorithmic content of Theorem 3 is implied by Theorem 1, and hence the main new content is in the existential statement. Before proving Theorem 3, we consider the special case of symmetric games which gives much of the intuition of what can and cannot be achieved.

3.3 Symmetric games

A multi-player game is *symmetric* if the payoff matrix of each player is symmetric (namely, permuting the identities of the remaining players keeps the payoff matrix unchanged), and moreover, the payoff matrices of any two players are identical to each other.

In this section G is a symmetric game, or more accurately, the *difference version* $\hat{G}$ of a symmetric game G. The payoff for P_i in $\hat{G}$ is computed as the payoff in G to P_i minus the payoff in G to P_0. Based on the symmetry of G and the 0-sum nature of $\hat{G}$, it follows that the product minimax value of all the players is 0. Thus, product-minimax-safety in $\hat{G}$ means obtaining a nonnegative expected payoff for every player.

The following proposition is a special case of Theorem 3.

Proposition 4. *For every symmetric game G, $\hat{G}_2$ is product-minimax-safe for M.*

Proof. For every entry in the payoff matrix G, add a constant to all payoffs to make the game 0-sum. It remains symmetric. Every symmetric game has a symmetric equilibrium [11]. Let M play this symmetric equilibrium. Then for I to play it as well is a mixed Nash, and the expected payoff for each player is 0, by symmetry. If I deviates from the mixed Nash, his expected payoff cannot increase, and hence the expected sum of payoffs of other player cannot decrease. By symmetry, no player controlled by M can then have negative expected payoff. This implies that the same strategy for M in $\hat{G}_2$ is product-minimax-safe. $\square$

Let us end this section with a few comments on what happens when G is symmetric, but I represents several players rather than just one. For concreteness, suppose that I represents two players (but the same principles can be generalized to a larger number of players).

If n, the number of players controlled by M, is divisible by 2, then the results extend in the following sense. M can partition his players into pairs, resulting in a new symmetric game with half the players (where each pair of players in G is a single player in the new game). Hence all results transfer, but with guarantees to pairs of players rather than to individual players. This can further be changed to a guarantee for an individual player by randomly permuting the order of players in a pair, and hence guaranteeing for each player the average of the guarantee for the pair. (In symmetric games, coalitions with nontransferable utilities can implement a transfer of utility to achieve fairness within coalition, by randomizing over the members).

If n is not divisible by 2, the guarantees become weaker. (Theorem 3 is still true as stated, but does not form an incentive for players to join the coalition formed by M.) Consider for example a 5-player symmetric game in which each player announces a number in $[m]$ (where m is large), and a player gets payoff of 1 if exactly one other player announced the same number as he did, and a payoff of 0 otherwise. In this game, a coalition M of three players cannot guarantee to its members an average expected payoff equal to the average of the remaining two players (that if controlled by I can coordinate to both report the same number in $[m]$, chosen at random).

3.4 Proof of safety for arbitrary games

Here we prove the existential part of Theorem 3. (The algorithmic part then follows from Theorem 1.)

Proof. Let v_i be the product minimax value of player P_i. As in the proof of Theorem 1, scale the payoff for each player P_i (for $i \in [n]$) by subtracting v_i, and let $\tilde{G}$ denote the obtained game. Showing that G_2 is product-minimax-safe for M is equivalent to showing that M has a strategy for $\tilde{G}_2$ that guarantees for each player P_i a nonnegative expected value.

Consider an arbitrary mixed Nash for $\tilde{G}'$ (the selection version of $\tilde{G}$). We claim that in this mixed Nash the expected payoff for every player P_i (for $i \in [n]$) is nonnegative. Equivalently, in this mixed Nash the expected payoff of every player P_i in G' is at least v_i. Assume otherwise, that there exists a player P_i ($i \in [n]$), whose expected payoff in G' is lower than v_i. Let x denote the mixed strategies for P_j ($j \in [n]$, $j \neq i$) as in the mixed Nash. Then P_i is *almost* playing the marginal game that remains between herself and P_0 when the remaining players play according to x, which has value at least v_i. The only difference is that P_i gets payoff only if P_0 chooses i. However, the choice of i is independent of the choice of x, and the strategy of P_0 on its first coordinate conditioned on having i on the second coordinate is a strategy for P_0 in the original game G. Hence the lower bound guarantee on expected payoff in G is transferred also to G' (otherwise P_i would not be in a Nash equilibrium in G' – he will have an incentive to change his mixed strategy).

We next claim that this implies that regardless of the strategy played by P_0, the expected sum of payoffs for P_i ($i \in [n]$) when playing according to this mixed Nash is nonnegative. Assume otherwise, that P_0 has a strategy for which the expected sum of payoffs for P_i (when playing according to the mixed Nash equilibrium) is negative. Then, the payoff for P_0 would be positive, which means that P_0 can improve its expected payoff from the mixed Nash equilibrium, in contradiction. This means that in the 0-sum game $\tilde{G}'_2$ (which is similar to $\tilde{G}'$, except that M controls players P_i for $i \in [n]$ and gets as payoff their sum of payoffs from $\tilde{G}'$), the strategy given above ensures M a nonnegative expected payoff.

Finally, we claim that using the same strategy in $\tilde{G}_2$ ensures a nonnegative payoff for every player P_i controlled by M. If this was not the case, say for a particular P_i, then I would have a strategy in $\tilde{G}'_2$ picking this particular i and achieving a positive payoff, implying a negative payoff for M in $\tilde{G}'_2$, reaching a contradiction.

□

3.5 A note on complexity

In [5] it is shown that computing the threat value (which is precisely the product minimax value) is NP-hard, even for symmetric games. This makes it difficult for M to deduce what are the product minimax values that it needs to attain for the players that it represents. Nevertheless, if these values are given to M as input, it can attain it via a polynomial time strategy, by Theorem 1.

Let us point out that for symmetric games, neither Proposition 4 nor its proof ensures for M that its vector of expected payoffs attains or exceeds the product minimax safety vector. Consider a 3-player symmetric game with two strategies, "cooperate" and "defect". If all players cooperate then all payoffs are 3. If one player defects his payoff is 0 and other payoffs are 1. If two or more players defect all payoffs are 2. The product minimax value for each player is strictly above 1. (Here is a simple way of achieving such a value. If one other player plays the pure strategy defect and the other plays the pure strategy cooperate, then the given player plays defect. In any other case, the given player cooperates.) However, the proof of Proposition 4 may (though also may not, as it has nondeterministic components) produce for M the strategy of always cooperating, which might (if I defects) lead to a payoff of only 1 to each of its players.

Nevertheless, despite the fact that it is NP-hard to compute the product minimax value of the players even in symmetric games, for the case of symmetric games, we have the following algorithmic results.

PROPOSITION 5. *There is a polytime strategy for M in symmetric games that ensures at least the product minimax value to each of its players.*

PROOF. By symmetry, the product minimax value for each player is the same. We denote this value by v. Consider now the 0-sum two player game between M and I is which M attempts to maximize the expected sum of payoffs for its players, and I tries to minimize it. The value of this game for M is at least nv (since we have shown that existentially M can guarantee each of its players an expected payoff of v), and perhaps larger. Let $V \geq nv$ denote this value. A mixed strategy for M that achieves value V can be computed in polynomial time (as this is a 0-sum game). Thereafter, this strategy can be further randomized by permuting the players controlled by M at random, ensuring (by symmetry) each one of the players expected payoff at least $V/n \geq v$. □

3.6 Additional safety notions in symmetric games

Recall that $\hat{G}$ denotes the difference version of a symmetric game G (where the payoff for P_i is computed as the payoff in G to P_i minus the payoff in G to P_0). The product minimax value of all the players in $\hat{G}$ is 0; thus, product-minimax-safety in $\hat{G}$ means obtaining a nonnegative expected payoff for every player. We can extend the notion of *product-minimax-safe*, and say that M *product-minimax-wins* a game if it can guarantee every player an expected value that is strictly greater than her product minimax value. Thus, M product-minimax-wins $\hat{G}_2$ if M has a mixed strategy that guarantees positive expected payoff for each one of its players.

Up to this point we considered the *vector version* of a master game, where the payoff of M is a vector of its players' payoffs. It will be instructive for us to consider also the *minimum version* of $\hat{G}_2$ in which the payoff for M is the minimum of the payoffs to all its players. (Note that expectation of minimum is never higher than minimum of expectation, and hence the minimum version is more difficult for M than the vector version.)

PROPOSITION 6. *There are symmetric games G for which M product-minimax-wins the corresponding game $\hat{G}_2$ even in the minimum version, and there are symmetric games G for which M product-minimax-wins the corresponding game $\hat{G}_2$ in the vector version, but not in the minimum version.*

PROOF. Consider G^+ with three players and two strategies (0 or 1) per player, where the payoff to a player is the number of other players playing his strategy. Then in the respective $\hat{G}_2^+$, M can play $(0,0)$ with probability 1/2 and $(1,1)$ with probability 1/2, achieving payoff 1/2 to each of its players. Here M achieves expected payoff of 1/2 even in the minimum version, thus product-minimax-wins even in the minimum version.

Alternatively, consider G^- where the payoff to a player is the number of other players not playing his strategy. Then in the respective $\hat{G}_2^-$, M can play $(0,1)$ with probability 1/2 and $(1,0)$ with probability 1/2. Here M product-minimax-wins in the vector version and the selection version, but not in the minimum version (where he can guarantee only 0). □

Some comments on Proposition 6 are in order. It shows that for some games (e.g., G^+) it is good for M to play identically for all its players, and for others (e.g., G^-) it is a mistake to do so. Also, observe that for G^+ and G^-, playing independently randomly and uniformly is product-minimax-safe for M (but not winning) in the vector version, but not product-minimax-safe in the minimum version.

The following proposition shows that the offers that can be guaranteed in the vector version (asserted in Proposition 4) do not extend to the minimum version.

PROPOSITION 7. *There are games for which M is not product-minimax-safe in the minimum version of the corresponding $\hat{G}_2$.*

PROOF. Let G be the following three player game (for large enough m). If two (or three) players play the same strategy they get payoff 0. For players that do not collide their payoff is computed as follows. Add the numbers (in $[m]$) reported by the players and consider the sum modulo 3. This determines which of the three player wins a payoff (the lowest report if the result is 0, second lowest if 1, highest if 2). Other players get no payoff. One of M's player cannot get payoff. Hence the minimum payoff for M is always 0. If I plays randomly (and m is large enough), it has probability roughly 1/3 to get a positive payoff. □

4. SAFETY GUARANTEES IN REPEATED GAMES

We consider here repeated games. We say that M approaches a safety vector $\bar{v}$ for its n players if regardless of the strategy of the independent player I, for every player i controlled by M, the expected sum of payoffs for player i over T rounds is $T\bar{v}_i - o(T)$. The nature of the safety vector $\bar{v}$ will be dependent on the context. We describe several settings of repeated games, and they differ by the information available to M.

4.1 Full information

Blackwell's approachability theorem [4] deals with approachability issues in general. One of its consequences is the following generalization of the minimax theorem to vector payoffs.

Consider a two player game between players Q_1 and Q_2 where the payoff for Q_1 is a vector. A vector $\bar{v}$ is *vector-minimax achievable* if for every mixed strategy of Q_2, player Q_1 has a mixed strategy with expected payoff at least $\bar{v}$. Vector $\bar{v}$ is *vector-maximin achievable* if Q_1 has a mixed strategy such that no matter what Q_2 plays, the expected payoff for Q_1 is at least $\bar{v}$. For vector payoffs the well known minimax theorem does not apply. That is, a vector that is vector-minimax achievable need not be vector-maximin achievable. However, Blackwell's approachability theorem implies that if a vector $\bar{v}$ is vector-minimax achievable, then it is also approachable (in a repeated game, where one considers the average payoff vector over all rounds). Moreover, this is an if and only if relation.

The above general result about games with vector payoffs is applicable in our setting, with M serving as Q_1, and I serving as Q_2. The master player M controls n players, and hence his payoff can be viewed as an n-dimensional vector (one coordinate for each player that M represents). In the one-shot game, the payoff vectors that are safe for M are precisely those that are vector-maximin achievable for Q_1. Under this interpretation, Theorem 3 identifies a particular class of vectors that are vector-maximin achievable, namely, those vectors that in each coordinate i have the product-minimax value for player P_i. Proposition 6 provides examples showing vectors that are strictly higher than the vector of product-minimax values, but nevertheless are vector-maximin achievable.

Blackwell's theorem implies that there are vectors that are not safe for M in the one shot game but are nevertheless approachable in the repeated games settings. Here is an explicit example that illustrates this. Consider a 3-player game with players P_0, P_1 and P_2. Only actions of P_0 and P_1 determine the payoffs of the game, and only P_1 and P_2 can receive any payoff. Each of the two players P_0 and P_1 has two possible actions, where one action gives P_1 a payoff of 1 and the other action gives P_2 a payoff of 1. The master player M controls P_1 and P_2, and the independent player I is P_0. The vector $\bar{v} = (1, 1)$ is not safe for M in the one shot game: whatever mixed strategy M has, there is one player (P_1 or P_2) for which P_1 gives expected payoff less than 1, and then I can give this player an additional payoff of 0, preventing it from achieving an expected payoff of 1. However, $\bar{v} = (1, 1)$ is approachable by M in the repeated setting. One could prove this fact using Blackwell's theorem, but for this simple game, one can see this directly: M plays arbitrarily in round 1, and in every round after that M plays (for P_1) the opposite of what I played in the previous round. Hence at every round t, either both P_1 and P_2 have accumulated a payoff of exactly t, or one of them accumulated a payoff of $t - 1$ and the other $t + 1$. As t grows, the average payoff vector converges to $(1, 1)$.

4.2 Only realized payoffs are observable

Here we consider a version in which the payoff matrices are not known to M. Moreover, they cannot be inferred from repeated play because the actions of I are not observable. All that M observes is the actual realized payoff vector after each round of play.

For symmetric games, a natural goal for M is to maximize the sum (over all n players) of payoffs, because randomizing over players spreads this sum evenly among them. This task of M in a repeated game can be cast as a so called *bandit problem*, for which solutions are known [6].

For nonsymmetric games the situation becomes more complicated, but M can still attain sublinear regret (for an appropriate choice of safety vector) using a combination of bandit algorithms and Blackwell's approachability theorem. This turns out to be a special case of a more general result (concerning expert and bandit algorithms in vector settings) that will be presented in a companion paper (in preparation), and hence will not be discussed further in the current paper.

4.3 Only actions are observable

Here we consider an *oblivious* version in which M does not know the payoffs, and never observes them, extending a setting of [7] from two players to multiple players. The game G is initially defined only in terms of sets of strategies available to the players. In the one round version the players M and I announce their strategies, and only after the game is played I announces a payoff matrix of its choice. In the repeated version the game is first played T times, and after all T repetitions are completed I gets to choose the payoff matrix. To give M hope of providing meaningful performance guarantees, we limit the choice of payoff matrix to be symmetric and the entries to be bounded independently of T (say by 1).

We shall use here the notion of *differential safety*; that is, as T tends to infinity, the difference between the expected average payoff of every player i over T rounds and that of I is $o(T)$.

THEOREM 8. *For every symmetric game G, M is differential safe in the oblivious repeated version. Moreover, a safe strategy for M can be found in time polynomial in m^n and T.*

PROOF. We combine the proof technique of [7] (that was previously used for $n = 1$) together with our Theorem 3.

We provide a strategy for the master player that guarantees for each player P_i under his control an expected average difference (over T periods) of

$$\mathbf{E}\left[\left|\frac{1}{T}\sum_{t=1}^{T} v_i^t\right|\right] \leq \sqrt{\frac{m}{T}}m^{n/2},$$

where v_i^t is the difference between the payoff of player P_i and player I in period t. Note that since the expectation is taken over the absolute value, this bounds actually serves as a lower bound on the payoff of player i.

In round t consider $nm^{(n-1)}$ auxiliary matrices $A^i_x[t]$, where index $i \in [n]$ specifies a player controlled by M, and index x specifies a choice of strategies for the other players controlled by M. Entry $A^i_x(j,k)[t]$ shows the difference in number of times P_i played k and I played j versus a play of (j,k) on periods $1,\ldots,t-1$, conditioned on x. Each matrix $A^i_x[t]$ is antisymmetric. A mixed strategy for M also implicity defines a distribution over x, and hence a distribution over auxiliary matrices. Define the "pretend" payoff matrix of P_i as the weighted average of the antisymmetric matrices $A^i_x[t]$, induced by the distribution over x. As a weighted average of antisymmetric matrices, the pretend payoff matrix is still antisymmetric. Being antisymmetric and 0-sum, P_i has a strategy for this payoff matrix with nonnegative expected payoff.

Using Theorem 3, M has a strategy that simultaneously gives expected nonnegative payoff to each of his players P_i in their respective pretend payoff matrices. Moreover, this strategy can be computed efficiently (as implied by Theorem 1). We claim that by playing such a strategy in every round (relative to the pretend payoff matrices of the respective round), M achieves the asserted bound for every player P_i.

Let $\phi_i = \sum_{t=1}^{T} v^t_i$. Then, $|\phi_i| \le \frac{1}{2}\sum_{x,j,k}\left|A^i_x(j,k)[T+1]\right|$. It holds that

$$\begin{aligned}(\mathbf{E}[|\phi_i|])^2 &\le \mathbf{E}[\phi_i^2]\\ &\le \mathbf{E}\left[\left(\sum_{x,j,k}\frac{1}{2}\left|A^i_x(j,k)[T+1]\right|\right)^2\right]\\ &\le \mathbf{E}\left[\frac{m^2m^{n-1}}{4}\sum_{x,j,k}\left(A^i_x(j,k)[T+1]\right)^2\right],\end{aligned}$$

where the last inequality follows by Cauchy-Schwartz.

Now let $\beta^i_t = \sum_{x,j,k}(A^i_x(j,k)[t])^2$. We claim that $\mathbf{E}\left[\beta^i_t\right] \le 2t$ for every t. To see this, let φ^i_t denote the expected "pretend" payoff of player P_i in period t, under the specified strategy of M. Simple algebra reveals that $\mathbf{E}\left[\beta^i_{t+1} - \beta^i_t\right] \le 2 - 2\varphi^i_t$. But since $\varphi^i_t \ge 0$, it follows that $\mathbf{E}\left[\beta^i_{t+1} - \beta^i_t\right] \le 2$. Combining the last inequality with $\beta^i_0 = 0$ implies that $\mathbf{E}\left[\beta^i_t\right] \le 2t$, as promised. It follows that $\mathbf{E}\left[|\phi_i|\right] \le \sqrt{mT}m^{n/2}$. Hence, the average number of deviations that I is expected to build against a single player over T periods is bounded by $\sqrt{\frac{m}{T}}m^{n/2}$. $\square$

5. CONCLUSION

One of the major challenges in the "agent perspective" to multi-agent systems is to deal with guarantees an agent can obtain, without assuming rationality of the other agents. This work pushes the envelope of that fundamental attempt by showing several results on the effective use of a master agent (aka mediator). Our two major results are:

- Every game admits a product-minimax-safe strategy for M — a strategy that guarantees for every player in M's coalition an expected value of at least her product minimax value (which is at least as high as her minimax value and is often higher). A player cannot guarantee for herself this value, when playing independently.

- In repeated symmetric games a master player who never observes a single payoff can guarantee (in an asymptotic sense) for each of its players a similar performance to that of the independent player, even if the latter gets to choose the payoff matrix after the fact.

These results are augmented with corresponding algorithmic results. The results expand on early foundational work in game theory, and work on mediators in AI.

6. REFERENCES

[1] R. Aumann. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1:67–96, 1974.

[2] R. Aumann. Acceptable Points in General Cooperative n-Person Games. *Contribution to the Theory of Games, Vol. IV, Annals of Mathematics Studies, 40, editors A.W. Tucker and R.D. Luce*, 287–324, 1959.

[3] R. Aumann. The Core of cooperative Games without Payments. *Transactions of the American Mathematical Society*, 1961.

[4] D. Blackwell. An Analog of the Minimax Theorem for Vector Payoffs. *Pacific Journal of Mathematics*, 6: 1–8, 1956.

[5] C. Borgs, J. Chayes, N. Immorlica, A. Kalai, V. Mirrokni, and C. Papadimitriou. The Myth of the Folk Theorem, Games and Economic Behavior, 2009. Conference version appeared in STOC 2008.

[6] N. Cesa-Bianchi and G. Lugosi. Prediction, Learning, and Games. Cambridge University press, 2006.

[7] M. Feldman, A. Kalai and M. Tennenholtz. Playing games without observing payoffs. *Innovations in computer science*, 106–110, 2010.

[8] D.B. Gillies. Solutions to general non-zero-sum games. *In A. W. Tucker and R. D. Luce, editors, Contributions to the theory of games, volume IV, pp. 47–85.* Princeton University Press, 1959.

[9] J. Hirshleifer. From weakest-link to best-shot: The voluntary provision of public goods. Public Choice, Vol. 41, Number 3, 1983.

[10] D. Monderer and M. Tennenholtz. Strong mediated equilibrium. *Artificial Intelligence, Vol. 173, Number 1, pp. 180–195*, 2009.

[11] J. Nash. Non-cooperative Games. *The Annals of Mathematics*, 54(2):286–295, 1951.

[12] O. Rozenfeld and M. Tennenholtz. Routing Mediators. *Proceedings of IJCAI*, pp. 1488–1493, 2007.

Game-theoretic Resource Allocation for Malicious Packet Detection in Computer Networks

Ondřej Vaněk[†], Zhengyu Yin[*], Manish Jain[*], Branislav Bošanský[†], Milind Tambe[*], Michal Pěchouček[†]
[†] Faculty of Electrical Engineering, Czech Technical University, Prague. Czech Republic.
{vanek,bosansky,pechoucek}@agents.fel.cvut.cz
[*] Computer Science Department, University of Southern California, Los Angeles, CA. USA.
{zhengyu.yin,manish.jain,tambe}@usc.edu

ABSTRACT

We study the problem of optimal resource allocation for packet selection and inspection to detect potential threats in large computer networks with multiple computers of differing importance. An attacker tries to harm these targets by sending malicious packets from multiple entry points of the network; the defender thus needs to optimally allocate her resources to maximize the probability of malicious packet detection under network latency constraints.

We formulate the problem as a graph-based security game with multiple resources of heterogeneous capabilities and propose a mathematical program for finding optimal solutions. We also propose GRANDE, a novel polynomial time algorithm that uses an approximated utility function to circumvent the limited scalability caused by the attacker's large strategy space and the non-linearity of the aforementioned mathematical program. GRANDE computes solutions with bounded error and scales up to problems of realistic sizes.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multi-agent Systems; C.2.0 [**Computer-Communication Networks**]: Security and Protection

General Terms

Algorithms, Security, Performance

Keywords

computer networks, security, game-theory, approximation algorithm, submodularity

1. INTRODUCTION

The problem of attacks on computer systems and corporate computer networks gets more pressing each year as the sophistication of the attacks increases together with the cost of their prevention. A number of intrusion detection and monitoring systems are being developed in order to increase the security of sensitive information, and many research works seek methods for optimizing the use of available security resources [9, 20]. One such countermeasure is conducting *deep packet inspections*, a method that periodically selects a subset of packets in a computer network for analysis. However, there is a cost associated with conducting a deep packet inspection: it leads to significant delays in the throughput of the network. Thus, the monitoring system works under a constraint of limited selection of a fraction of all packets which can be inspected to bound the total delay.

Game-theoretic methods are appropriate for modeling such problems and the optimal behavior of the involved parties can be found using the well-defined concept of an equilibrium computation. We formulate this problem as a game between two players: the *attacker* (or the *intruder*), and the *defender* (the *detection system*). The intruder wants to gain control over (or to disable) a valuable computer in the network by scanning the network, compromising a more vulnerable system, and/or gaining access to further devices on the computer network. The actions of the attacker can therefore be seen as sending malicious packets from a controlled computer (termed *source*) to a single or multiple vulnerable computers (termed *targets*). The objective of the defender is to prevent the intruder from succeeding by selecting the packets for inspection, identifying the attacker, and subsequently thwarting the attack. However, packet inspections cause unwanted latency and hence the defender has to decide where and how frequently to inspect network traffic in order to maximize the probability of a successful malicious packet detection.

We build on previous work on game-theoretic approaches to network security [1, 11] and security games [9, 20] to present a novel approach to the challenges of malicious packet detection. In our approach, we follow the deep packet inspection scenario on an arbitrary network topology, and consider the following assumptions that hold true in this domain: the attacker can access the computer network through multiple entry points, can attack multiple targets of differing importance in parallel, and the defender has limited resources that can be used for packet analysis. To the best of our knowledge, there is no previous work considering together all of these aspects of the problem.

This paper offers following contributions: (1) we propose a novel game-theoretic model that can be characterized as a graph-based security game with multiple heterogeneous attacker's and defender's resources; (2) we give a mathematical program for finding the optimal solution for this problem

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

formulated both as a non-zero sum and zero sum game approximation; (3) we describe a polynomial approximation algorithm GRANDE (*GReedy Algorithm for Network DEfense*) that benefits from the submodularity property of the discretized zero-sum variant of the game and finds solutions with bounded error in polynomial time; (4) we experimentally evaluate these three algorithms and show the trade-off in computational time and quality of found solutions.

2. RELATED WORK

Game theory has been applied to a wide range of security problems, with many deployed applications in transportation networks [10, 20]. In fact, game-theoretic models of real-world security problems are applicable in a wide variety of domains with similar attributes, including (1) intelligent players, (2) varying preferences among targets, (3) and limited resources to protect targets. This has led researchers to model computer network security in game-theoretic frameworks and a large body of work has been created (summarized e.g. in [15]).

Most related is the recent work by Kodialam and Lakshman [11] since they also look at a scenario where the defender conducts inspections on possible paths from a source to a target. However, they look at a zero-sum setting for a single source and a single target. Similarly, Otrok et al. [17] present solutions for a domain with a single target, where the attacker potentially uses multiple packets for an attack. On the other hand, Chen et al. [2] present solutions for heterogeneous targets, with multiple attacker's resources. However, they only consider detection at the target nodes.

From the research focused on the security-games models, Korzhyk et al. [12] present a polynomial algorithm for general-sum security games with multiple attacker's resources, however, without constraining underlying graph structure. Jain et al. [9] present an algorithm for securing an urban network with many sources and heterogeneous targets. However, this model is zero-sum and the attacker has a single resource. Our approach mainly differs in considering a network-security domain, where the payoffs are not necessarily zero-sum and player's utilities have more complex structure. We also model the attacker with multiple resources used in parallel, so the defender succeeds in preventing an attack only if all the attacker's paths leading to a single target are intercepted.

Our work also exploits the submodular properties of the network security domain. Submodular functions for optimal resource allocation optimization in adversarial environments were first introduced by Freud et al. [6], and further developed by Krause et. al [13]. However, they do not work with continuous defender's resources and consider only zero-sum setting.

3. FORMAL MODEL

3.1 Environment

We define the problem of the packet selection for inspection as a two-player game between *the attacker* and *the defender*. The game is played on a graph $G(N, E)$ that represents the topology of a computer network. The set of nodes can be decomposed into three non-empty sets: (1) S is the set of nodes that can be under the control of the attacker; (2) T is the set of targets ($S \cap T = \emptyset$); (3) $A = N \backslash \{S \cup T\}$ is the set of all other nodes in the network. From A, the defender can inspect the traffic only on a subset of *intermediate* nodes $I \subseteq A$ (representing, for example, firewalls, proxy servers, etc.). For our problem, we consider only nodes from S, T, I.

The packages are routed in the network by an underlying *deterministic* routing protocol that is not under the attacker's control; therefore, for each tuple $(s, t) : s \subseteq S, t \subseteq T$, there is either a fixed *single* path through intermediate nodes I, or there is a path without intermediate nodes leading from s to t, or there is no path from s to t in the graph. Thus, the defender does not need to consider allocation of resources to such intermediate nodes which do not lie on any path (given the set of sources and targets).

Each target t has an associated value $\tau_t \geq 0$ that represents the importance of the target; the defender loses τ_t if t is attacked successfully and gains 0 if she succeeds in preventing the attack. The attacker gains τ_t if t is attacked successfully. In case of an unsuccessful attack (i.e. a malicious packet was detected by the defender), the attacker pays a detection penalty $\gamma_s \geq 0$ associated with using the source s[1]. The penalty for the attacker models situations when the defender detects that source s is being used to send malicious packets and blocks this source from the network. Due to this penalty, the attacker may choose not to attack any of the targets, we thus define a virtual node in the network — a *dummy target* t_D that is directly connected to all sources, and for which $\tau_{t_D} = 0$. Finally, for each intermediate node n_i we define *flow* f_i that represents the amount of legitimate network traffic going through n_i. We assume this amount to be constant in time[2] and we assume that the amount of malicious packets sent by the attacker is fractional compared to the legitimate network traffic[3].

3.2 Players

Both players have multiple resources that they can use. The resources of the attacker are determined by the size of the set of sources $|S| = k$ and we assume that the attacker attacks from one source only a single target (however, one target can be attacked from multiple sources). The strategy of the attacker is to select k tuples determining which target will be attacked from each of the sources:

$$P = \{(s_i, t_i) : s_i \in S, t_i \in T, i = 1 \ldots k, \forall_{j \neq i} s_i \neq s_j\}$$

Since there is at most one path from source s to target t in the graph, we refer to the attacker's strategies as paths and denote them as $p_{(s,t)} \subseteq I$ with subscript omitted if the source-target is clear from the context. Hence, we denote $\mathcal{C}_t$ to be a set of such paths that originate in some source and end in a single target t. The attacker's resources are thus heterogeneous, since it may not be possible to attack the same set of targets from all sources.

The strategy of the defender is an assignment of a vector of probabilities $X = (x_1, x_2, \ldots, x_m)$, where for each intermediate node $n \in I$; ($m = |I|$), the probability x_i represents the fraction of the traffic going through the node n_i that will

[1]We assume standard IP packets with a source IP address, i.e. the source is traceable from the packet header; for *spoofing* attacks [8], γ_s is set to zero.

[2]If the amount of traffic varies in time (e.g., weekends vs. workdays, days vs. nights), we can compute multiple strategies and switch between them.

[3]For flooding attacks, different detection/prevention countermeasures instead of deep packet inspection are used.

be inspected. Therefore, the value x_i also represents the probability of the detection of a malicious packet sent by the attacker through node n_i[4]. The available amount of the defender's resources is determined by the maximum amount of inspected traffic B — the maximum allowed average latency in the computer network. Therefore, the defender is seeking her strategy satisfying the following constraint:

$$L(X) = \sum_{n_i \in I} x_i \cdot f_i \leq B$$

where $x_i \cdot f_i$ represents the expected number of packets that were inspected at node n_i (for the complete network, we use $L(X)$ for brevity)[5]. As in the case of the attacker, the heterogeneity of the defender's resources is given by the structure of the graph (different intermediate nodes provide a malicious packet detection for different groups of targets) and different flows for different $n_i \in I$.

Finally, when designing an intrusion detection system, a typical assumption is that the attacker will have a full knowledge of techniques used by the system [19] and together with the full knowledge of the network structure[6] the attacker is able to reconstruct the defender's strategy; the attacker is thus assumed to know the probability with which a packet may be inspected at each of the intermediate nodes. In this paper, we thus assume Stackelberg game formulation; however, the relaxation of these assumptions is subject of further research.

3.3 Utility Functions

The utility functions of both the players are a function of the probability of detection of the sent malicious packets. Since the intermediate nodes inspect packets independently, the probability of a single malicious packet **avoiding detection** along the path p is given by:

$$\pi(X, p) = \prod_{i \in p} (1 - x_i) \qquad (1)$$

where X is a strategy of the defender in the form of allocation of detection resources at nodes n_i. The probability of **detecting** a packet on each path for a set of paths $\mathcal{C}$ is computed as:

$$\psi(X, \mathcal{C}) = \prod_{p \in \mathcal{C}} [1 - \pi(X, p)]$$

Now, if P denotes the strategy of the attacker (i.e., paths p_j in the graph), and $\mathcal{C}_t$ is the set of all paths chosen by the attacker leading to a target t, the utility of the defender $U_d(X, P)$ is defined as follows:

$$U_d(X, P) = -\sum_{\forall t \in T} \tau_t \cdot [1 - \psi(X, \mathcal{C}_t)] \qquad (2)$$

where the term $(1 - \psi(X, \mathcal{C}_t))$ denotes the probability that at least one malicious packet avoids detection and reaches target t. Therefore, the defender's utility is an expected loss of values of targets that were reached by malicious packets.

[4]This assumes having a perfect detector.

[5]The latency is not computed for every path but averaged over the entire network, keeping our model tractable.

[6]Using standard network analysis tools, such as Nmap.

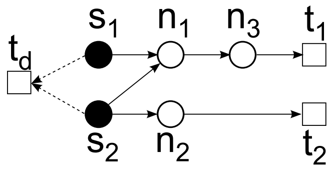


Figure 1: Example graph. Two source nodes s_1 and s_2, three intermediate nodes n_1, n_2 and n_3, two target nodes t_1 and t_2, and a dummy target node t_d.

Analogously, we define the attacker's utility $U_a(X, P)$ as:

$$U_a(X, P) = -U_d(X, P) - \sum_{p_{(s,t)} \in P} \gamma_s \cdot [1 - \pi(X, p)] \qquad (3)$$

The attacker's utility equals to the expected gain of values of targets that were reached by malicious packets reduced by the detection penalty γ_s for each path that the attacker uses; recall the attacker needs to pay a penalty when a packet is detected, as discussed in Section 3.1. As such, for any non-zero γ_s, the game is not zero-sum.

3.4 Example

The example on Figure 1 depicts a simple graph with two sources s_1, s_2, three intermediate nodes n_1, n_2, n_3 with flows $f_1 = 5, f_2 = 3, f_3 = 5$ and two targets t_1, t_2 with values $\tau_{t_1} = 2, \tau_{t_2} = 6$. The number of adversary resources $k = 2$ and defender's latency budget is set to $B = 6$. The attacker's strategy set is:

$$\begin{aligned}\mathbb{P} = \{ \quad & [(s_1, t_1), (s_2, t_1)], [(s_1, t_1), (s_2, t_2)], \\ & [(s_1, t_D), (s_2, t_1)], [(s_1, t_D), (s_2, t_2)], \\ & [(s_1, t_1), (s_2, t_D)], [(s_1, t_D), (s_2, t_D)]\}.\end{aligned}$$

If, for example, the defender chooses her strategy to be $X = \{x_1 = 0.0, x_2 = 0.5, x_3 = 0.1\}$, the latency caused is $L(X) = 0.0 \cdot 5 + 0.5 \cdot 3 + 0.1 \cdot 5 = 2$. If the attacker selects a strategy $P = [(s_1, t_1), (s_2, t_1)]$, the defender's utility will be: $U_d(X, P) = -2 \cdot [1 - (1 - (1 - 0.0) \cdot (1 - 0.1)) \cdot (1 - (1 - 0.0) \cdot (1 - 0.1))] = -1.98$. The attacker's utility will be (when setting $\gamma_{s_1} = \gamma_{s_2} = 1$): $U_a(X, P) = -U_d(X, P) - 1 \cdot (1 - (1 - 0.0) \cdot (1 - 0.1)) + (1 - (1 - 0.0) \cdot (1 - 0.1)) = 1.98 - 0.2 = 1.78$. The optimal setting is $X^* = \{x_1 = 0.0, x_2 = 0.857, x_3 = 0.686\}$, forcing the attacker to select $P^* = [(s_1, t_D), (s_2, t_2)]$, giving the defender expected utility $U_d(X^*, P^*) = -0.858$; and the attacker's expected utility is $U_a(X^*, P^*) = 0.001$.

4. SOLUTION APPROACH

First, we look for Strong Stackelberg Equilibrium (SSE) of the full general-sum game. Second, we propose a zero-sum game model (Section 4.2) that is capable of scaling to larger problem sizes. Third, we prove the submodularity of the problem (Section 4.3). Finally, we propose GRANDE, an iterative algorithm for finding suboptimal solutions in polynomial time (Section 4.4).

4.1 General-sum Game Model

Given the assumptions stated above, we model the problem as a Stackelberg general-sum game between the defender and the attacker: the defender is the leader, committing to her strategy first, and the attacker is the follower, choosing his strategy after the leader's commitment. The SSE gives the optimal strategy for the leader given that the follower acts with the knowledge of this optimal leader strategy. It is found by solving multiple programs [3] as follows:

$$\max_X \quad U_d(X, P^*) \tag{4}$$
$$\text{s.t.} \quad L(X) \leq B \tag{5}$$
$$U_a(X, P^*) \geq U_a(X, P) \quad \forall P \tag{6}$$
$$x_i \in [0, 1] \tag{7}$$

The inputs of the programs are all possible pure strategies of the attacker P and P^* is assumed to be the current best response for the attacker. We compute the defender's strategy X that maximizes the defender's utility $U_d(X, P^*)$ (Equation 4) while adhering to the latency constraint (Equation 5) and ensuring that the assumed best response of the attacker P^* is better than all other attacker's pure strategies $\forall P$ (Equation 6). While this program may not always be feasible if some choice of P^* is strictly dominated by others, it will still always return a solution for all non-dominated P^*. The number of programs needed to be solved to find an optimal solution is given by the number of attacker's strategies, which is $|T|^k$, since there are $|T|$ targets and k sources. This approach has two main scalability limitations: first, the non-linear formulations of U_d and U_a prohibit us from using fast linear-program solvers; second, the attacker's strategy space is extremely large (for a graph with 5 sources, 5 targets and one *dummy* target, we get over 7500 (6^5) programs with similar number of non-linear equations), limiting the usability of the non-linear solvers.

An alternative approach, inspired by algorithms computing SSE by solving a single mixed-integer program [18], would introduce into each Equation 6 an integer variable z_i (for each attacker's strategy P_i) and restrict the variables by $\sum z_i = 1$, i.e., only one attacker's strategy can be selected as the best response. However, this program would be very large, having $(|T|^k)^2$ non-linear equations (which is over 56 million for the problem with 5 sources and 5 targets). Hence, we look at the zero-sum game formulation for the problem which allows us to exploit the structure in ways that keep the solution tractable.

4.2 Zero-sum Game Approximation

Finding an optimal solution using the full general-sum game representation is computationally demanding on large problems. We thus propose a zero-sum game formulation which reduces the complexity of the model. Setting the cost of each source to $\gamma_s = 0$, the utility function of the attacker becomes a negation of the utility of the defender ($U_a(X, P) = -U_d(X, P)$), and the game becomes zero-sum. In zero-sum games, the SSE is also a Nash Equilibrium, which can be computed using the minimax theorem. This approximation causes an error quantified in Section 5. SSE of our zero-sum game can be found by solving a single non-linear mathematical program:

$$\max_X \quad \mathcal{V} \tag{8}$$
$$\text{s.t.} \quad U_d(X, P) \geq \mathcal{V} \quad \forall P \tag{9}$$
$$L(X) \leq B \tag{10}$$
$$x_i \in [0, 1] \tag{11}$$

In this mathematical program, the main scalability limitation persists – as for the general sum model – the non-linear nature of the utility function (Equation 9) and the size of the linear program, depending on the size of the attacker's strategy space (Equation 9). However, in spite of the large problem size, zero-sum games are generally easier to solve optimally (e.g., iterative algorithms can be used as in [7, 9]) or to approximate [14]. We follow the latter approach and investigate approximation algorithms that utilize the property of submodularity and are able to find solutions for zero-sum games with guaranteed bounded error.

4.3 Submodularity

In our problem formulation, the defender's resources exhibit diminishing returns, i.e., as the number of defender's resources is increased, the marginal utility of deploying one extra resource keeps decreasing. This property is formalized by the concept of submodularity [16] which is utilized in many domains (e.g., sensor networks) to design effective algorithms for solving problems with a large number of defender's resources. A real-valued function F defined on subsets A of a finite set V is called submodular, if for all $A \subset B \subset V$ and for all $s \in V \backslash B$ holds that $F(A \cup \{s\}) - F(A) \geq F(B \cup \{s\}) - F(B)$. The constrained optimization of a submodular set function is NP-hard in general, however, a number of approximation algorithms with provable quality guarantees can be used [22].

In our formulation, we have intermediate nodes that detect the activity of the attacker. The value of the detected activity in this problem setting is a probability between $[0, 1]$, as opposed to being binary which is generally assumed in submodularity. Thus, our requirements do not meet the assumptions of most prior work on submodularity, except the work by Vondrak et al. [22], which studied smooth continuous extension of submodular functions by taking expectations, defining sensors making observations independently with probability in range $[0, 1]$. The approach requires the continuous function to be twice partially differentiable and an approximation bound is established by exploiting the up-concavity[7] of the resulting continuous function [5]. However, this work is not applicable to our problem since our objective function, $U_d(X, P)$, is not up-concave which can be determined by taking the double derivative of the defender's utility function.

4.4 GRANDE Algorithm

We choose a different approach (in contrast to standard submodular approaches) to exploit the submodularity of the problem: we transform U_d into a submodular function defined over sets by discretizing the sampling rate of each node and we allow nodes to sample only a fixed portions of traffic defined by a discretization step $d \leq 1$; e.g. for $d = 0.1$, the sampling rate at each node can be set only to $0, 10, 20, \ldots, 100\%$. Then, each node n_i can be seen as a set of $1/d$ *sensors* $\mathcal{S}(n_i) = \{n_i^1, n_i^2, \ldots, n_i^{1/d}\}$. A sensor n_i^j can be switched either on (and sample a portion of the traffic) or off which is expressed by a binary variable $x_i^j \in \{0, 1\}$ having value of 1 for a sensor switched on. The defender's strategy is defined using the sensor notation as $X = \{x_i^j\}$. We redefine the Equation 1 defining the probability of a sin-

[7]Up-concavity means that the function is concave along any non-negative direction vector; however, it is not necessarily concave in all directions.

Algorithm 1 *GReedy Algorithm for Network DEfense.*

```
$budget \leftarrow B$
$I \leftarrow nodesOnPaths$
repeat
    $updated \leftarrow$ false
    $bestNode \leftarrow$ null
    $bestIncrement \leftarrow 0$
    $attackerBR \leftarrow getAttackerBR(graph)$
    for $node \in I$ do
        $increment = getSecurityIncrement(d, attackerBR)$
        if $bestIncrement < increment$ then
            if $d \cdot flow(node) < Budget$ then
                $bestIncrement \leftarrow increment$
                $bestNode \leftarrow node$
            end if
        end if
    end for
    if $bestNode != $ null then
        $bestNode.sampling \leftarrow bestNode.sampling + d$
        $budget \leftarrow budget - d \cdot flow(node)$
        $updated \leftarrow$ true
    end if
until not $updated$
```

gle malicious packet avoiding detection along a path p as:

$$\pi(X,p) = \prod_{n_i \in p} (1 - d \cdot \sum_{\mathcal{S}(n_i)} x_i^j) \tag{12}$$

Having a submodular utility function defined over sets for the defender U_d (which has the same formulation as in Equation 2), we are able to design an iterative greedy algorithm to achieve at least $(1 - 1/e)$-optimal (approximately 63.2%) solution (compared to the zero-sum game SSE) [6] similarly to work of Krause et al. [14]. However, it is also necessary to consider the cost of each sensor, given by the budget constraint $L(X) \leq B$. When inspecting the same ratio of packets at two nodes n_i, n_k with flows $f(n_i) > f(n_k)$, the cost of inspection at the node n_i (and thus switching on a sensor at node n_i) is higher than inspection cost at node n_k. The cost of switching on a sensor is defined as $c(n_i^j) = d \cdot f(n_i)$. As shown in [13], the greedy algorithm has to select a sensor x_i^{j*} with the highest cost-benefit ratio to guarantee bounded error of the solution:

$$x_i^{j*} = \arg\max_{x_i^j} \frac{U_d(X \cup \{x_i^j\}, P) - U_d(X, P)}{c(n_i^j)} \tag{13}$$

Based on this formalization, we introduce GRANDE (*GReedy Algorithm for Network DEfense*), depicted in Algorithm 1. GRANDE iteratively selects sensors with the highest security increment vs. cost ratio to add to the defender's strategy, following the greedy approach. To find the best sensor to add, we find attacker's optimal strategy *attackerBR* and test each candidate sensor against this strategy. The algorithm ends when there is no budget left or there is no sensor to be added.

The complexity of the algorithm is thus dependent on the number of nodes $n = |I|$, the discretization step d, the minimum amount of traffic flow at each node f, the sampling budget B and the complexity of the attacker's best response oracle $O(\mathcal{BR})$, and is $O(nB/fd) \cdot O(\mathcal{BR})$.

The attacker's optimal strategy *attackerBR* is a best response to the current defender's strategy (following the original Stackelberg formulation). The algorithm thus needs a fast best response oracle providing best response to the current strategy of the other player. The following section defines such oracle and provides insight into the complexity of this approach.

Algorithm 2 Attacker's Best Response Oracle

```
$H \leftarrow \{\}$
$K \leftarrow attackerResources$
$Pairs \leftarrow enumerateAllPairs()$
repeat
    $(s^*, t^*) \leftarrow emptyPair$
    for $(s,t) \in Pairs$ do
        if $U((s,t)|H) > U((s^*,t^*)|H)$ then
            $(s^*, t^*) \leftarrow (s,t)$
        end if
    end for
    $H \leftarrow H \cup (s^*, t^*)$
until $size(H) = K$
```

Attacker's Best Response Oracle

Recall that the attacker only selects for each source a target to attack and the routing path is automatically assigned. The attacker's best response is thus an optimal assignment of a target to every source, given a fixed defender's strategy X, maximizing the attacker's utility. The attacker's best response can be found using an iterative greedy approach.

Let's assume we have the defender's strategy — a mixture of sampling probabilities x_i for each node n_i. We can compute for each source-target pair, what is the likelihood of being detected $\rho_s^t = 1 - \pi(X, p_{(s,t)})$. Let's denote the source-target pairs by $STP = \{(s_1, t_1), \ldots, (s_n, t_m)\}$. We also know that two different source-target pairs (with different sources) can share the same target t. Given input $\{STP, \rho_s^t\}$, the best response is k source-target pairs, $\{(s_1,t_1),(s_2,t_2),\ldots,(s_k,t_k)\}$ such that the attacker's utility is maximized.

The greedy algorithm (summarized in Algorithm 2) works as the following: we choose one source-target pair at a time that maximizes the attacker's immediate gain in utility. Let's assume some pairs H, have been chosen and we need to choose the next one. Since pairs in H have already been chosen, we know there is some probability of successfully attacking a target t, denoted by q^t, which may or may not be 0. If we choose source s, and target t, the additional utility we will get will be:

$$\begin{aligned} U((s,t)|H) &= [1 - \rho_s^t \cdot (1 - q^t)] \cdot \tau_t - q^t \cdot \tau_t \\ &= (1 - \rho_s^t) \cdot (1 - q^t) \cdot \tau^t \end{aligned} \tag{14}$$

If H is empty, all $q^t = 0$ and $U((s,t)|\{\}) = (1 - \rho_s^t) \cdot \tau^t$, is the expected value of attacking t from s. The greedy algorithm would then choose (s^*, t^*) such that $U((s^*,t^*)|H)$ is maximized.

THEOREM 1. *The attacker's oracle always returns attacker's best response to a given defender's strategy.*

PROOF. Consider at any point of the algorithm, a set of source-target pairs H has been chosen. The greedy algorithm returns (s^*, t^*). We want to show (s^*, t^*) must be in the best solution conditioned on H being included. This will allow us to do induction on the number of pairs chosen. Let's denote the optimal solution by C^*. The first pair chosen, which is the best one from STP, must be in C^* because H_1 is empty (no condition required). And if the pairs up to k are all in the optimal solution, implying H_k is in C^*, therefore the $k+1$-th pair must be in the optimal solution.

This implies that we want to show (s^*, t^*) must be in the best solution conditioned on H being included. To show this by contradiction, we consider another candidate best solution C (having H) which does not have (s^*, t^*). Two cases to consider:

1. C contains no pair attacking target t^* other than those in H. Then we find an arbitrary pair (s', t') in C but not in H (such set is denoted as $C \backslash H$) and replace it by (s^*, t^*). We know the attacker gains exactly $U((s^*, t^*)|H)$ (since no other pair in $C \backslash H$ attacks t^*) and loses at most $U((s', t')|H)$ (since there might be another pair in $C \backslash H$). Recall $U((s^*, t^*)|H) \geq U((s', t')|H)$ given how (s^*, t^*) is chosen, the new solution $C + (s^*, t^*) - (s', t')$ must be better than C which also includes H, leading to a contradiction.

2. $C \backslash H$ has at least another pair attacking t^* that is not (s^*, t^*). Let the pair be (s', t^*). We replace it by (s^*, t^*). We know $\rho_{s^*}^{t^*} \geq \rho_{s'}^{t^*}$ because $U((s^*, t^*)|H) \geq U((s', t^*)|H)$. Therefore the total probability of successfully attacking t^* must increase after the replacing given other pairs in C remain fixed. Again this shows, $C + (s^*, t^*) - (s', t^*)$ is a better solution which is the contradiction.

Having reached contradiction in both points, we have shown that (s^*, t^*) must be in the best solution conditioned on H being included, implying validity of the induction step. $\square$

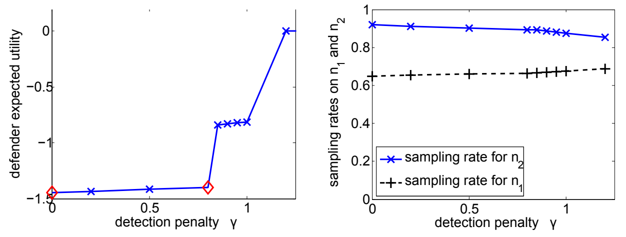
The complexity of the algorithm is $O(STn + S^2T) = O(n^3)$, where the $O(STn)$ is complexity of the initialization and $O(S^2T)$ is complexity of iterations. Here, S is the number of sources, T is number of targets and $n = |I|$ is the number of nodes.

5. EVALUATION

In the evaluation, we focus on exploring the trade-off between scalability and the quality of the solution. We consider the solution of the general-sum model to be optimal and compare it with the solution of the mathematical program representing the zero-sum game model, and the solution from GRANDE. Additionally, we want to explore finer properties of GRANDE, specifically, the dependency of the solution error on the discretization step of the sampling rate.

Experimental scenarios of the analyzed problem depend on a large set of parameters that affect both the performance of the algorithms, as well as the quality of produced solutions for the approximative ones. The key parameter is the graph on which the game is played; more specifically the number of intermediate nodes $|I|$, the number of sources $|S|$, and the number of targets $|T|$. Moreover, the degree of overlapping paths also plays an important role in the non-linear models. The detection penalty γ_s has no direct impact on the run time of GRANDE; the defender's budget B, traffic flow f_i and discretization step d proportionally influence mainly the run time of GRANDE.

While we conducted experiments for different graph structures, we present results only on *scale-free graphs* since these graphs are known to be the closest to general computer networks in their structure. We performed experiments with



(a) Defender's exp. utility. Red diamonds denote locally optimal solution.
(b) Distribution of defender's resources between n_1 and n_2 (log x axis).

Figure 2: Impact of detection penalty γ on the solution structure. While increasing value of γ, the defender redistributes her resources between n_1 and n_2 and her exp. utility changes (b), however, it stays equal to zero for $\gamma > 1.2$ (a).

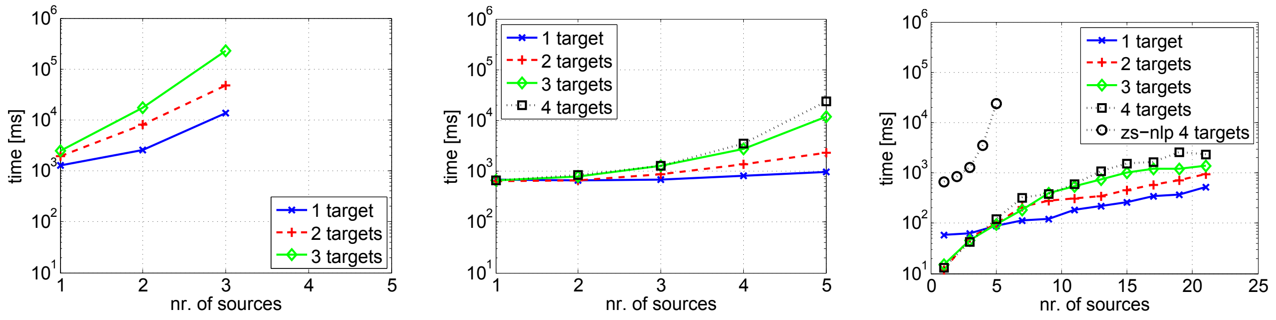
random flows (e.g. the flow at each node is set independently on the flow of the others) as well as with network-flow constrained traffic distribution (the flow at each node is computed from the network-flow equations by randomly selecting traffic sources and sinks in the network) which did not directly influence both the performance and quality of the solution. Without loss of generality, in every experiment, the traffic flow in the graph is set between $[0, 1]$ at each node. We have included the dummy target in each model to keep the graph size constant for all algorithms, even though the zero-sum model as well as the iterative algorithm never consider the attacker to attack the dummy target. The detection penalty was set to $\gamma_s = 1$ for each source.

5.1 Solving Non-linear Constrained Programs

To obtain an optimal solution of the program representing the general-sum game model, we use a non-linear solver to find optimal or locally optimal solution. NEOS server [4] provides on-line solvers for solving non-linear programs. We used LINDOGlobal [21], a non-linear constrained program solver able to find globally optimal solutions for many constrained non-linear programs. The input to the solver is a file describing the program in the GAMS format, which is sent by a remote procedure call to the NEOS server using XML remote procedure call API. The solution is computed on the server and the results are sent back to the user.

5.2 General-sum vs. Zero-sum Model

As a first step, we compare the quality of the general-sum and the zero-sum game model. The difference in the solution quality between these two models will be directly affected by the value of the detection penalty γ, as it can be observed from Equations 2 and 3. For the example described in Section 3.4, the trend of defender's expected utility while varying γ is depicted on Figure 2a. As γ is increased (i.e. the attacker is penalized more for a detected malicious packet, thus the utility of players is further from zero-sum), the defender's expected utility rises. Two rapid transitions occur for ($\gamma = 0.8$ and $\gamma = 1.2$) which are caused by the switch of attacker's strategies to attack the dummy target t_D. In the interval from $[0, 0.8]$, the attacker attacks from both sources, in the interval from $[0.8, 1.2]$ the attacker attacks only from one source, and from $[1.2, \infty]$, the attacker chooses not to attack at all. Figure 2b shows the distribution of defender's



(a) General-sum NLP (b) Zero-sum NLP (c) GRANDE

Figure 3: Scalability of the three models with respect to the number of sources and targets on scale-free graphs with 100 intermediate nodes (note the different scale of the x axis). For comparison of GRANDE with other two approaches, we have displayed one result of the zero-sum NLP in the Figure (c) denoting performance of the zero-sum model for 4 targets.

resources between nodes n_1 and n_2 as γ is varied. Notice that the defender has to redistribute the resources to discourage the attacker from attacking, while still adhering to the latency constraints, i.e. the node sampling rates multiplied by the flows through the nodes $x_1 \cdot f_1 + x_2 \cdot f_2 \leq B = 6$. The results of the zero-sum model are equal to the results of the general-sum model with $\gamma = 0$.

5.3 Performance

Figure 3 depicts the scalability of the three main algorithms. The results for general-sum NLP are restricted to a maximum of 3 sources and 3 targets, since the NEOS server limits the size of the input that can be sent to it. Similarly, for the zero-sum NLP, the limit was reached at 5 sources and 4 targets. However, even on these problem sizes it is possible to see performance trends: the runtime of the general-sum as well as zero-sum NLP is exponential (even in logarithmic coordinates) in the number of sources (i.e. number of attacker's strategies) and time needed to solve a graph with 3 sources and 3 targets is over three minutes in average for the general-sum NLP. Using the zero-sum NLP, we are able to compute solutions on graphs with 5 sources and 4 targets in approximately 30 seconds.

Comparing GRANDE to mathematical program formulations, we can observe its superiority on Figure 3c (performance of the zero-sum NLP on 4 targets is depicted as a single line on the left of the chart). The performance of GRANDE is linearly dependent on the discretization step d (see Section 4.4). The algorithm is able to find solution on graphs with 20 sources and 4 targets in seconds, having the discretization step set to $d = 0.01$ (which is sufficient to compute solutions with an average error under 10%, see Section 5.4). The largest problem tried, with 2000 nodes, 200 sources and 20 targets was solved in approximately 50 hours on a standard PC.

5.4 GRANDE Solution Error

The theoretical error bounds of greedy algorithms optimizing submodular set functions shown in [22] are valid only for zero-sum settings. We explore the error of GRANDE compared to the general-sum game solution, which can be possibly unbounded. It is necessary to set the discretization step of GRANDE to a specific step, which has a direct impact on the quality of solution. To evaluate the error, we have varied both the discretization step as well as attacker loss expressed by γ. The budget constraint of the defender was fixed to $B = 4$.

For every graph, we have computed the defender's resource allocation X^* and attacker's best response P^* using the program of the general-sum NLP, which served as a reference optimal solution (even if only a local optimum was found by the solver, due to lack of other globally optimal techniques). Then we computed the defender's resource allocation X^G using GRANDE. To evaluate the quality of X^G, we have found the attacker's best response P^G to X^G using the general-sum utility formulation. Then, we computed the error of X^G as $err = \frac{U_D(X^*,P^*)-U_D(X^G,P^G)}{\mathcal{T}}$, where $\mathcal{T} = \sum \tau_t$ (maximum achievable error).

Figure 4 quantifies errors of GRANDE from 50 different scale-free graphs with two sources and two targets (problem sizes limited due to restrictions imposed by the NEOS server). The graph depicts the median error (denoted by the circle) with the 25th and the 75th percentile (denoted by a thick bar) and maximal and minimal error (denoted by whiskers). As we refine the discretization step from 1 to 0.001 (i.e. the sensors can increase their sampling rate by 0.1% for $d = 0.001$), the quality of solution increases. An average error under 10% is reached when the discretization step is set to $d = 0.01$, however GRANDE is able to compute strategies with discretization step set to 0.001 resulting into errors under 5%. The variance is observed to be the highest for $d = 0.1$ as the solutions varied from close-to-optimal to 100% ineffective.

6. CONCLUSION

Effectively securing large computer networks without stifling the quality of service is a practical and theoretical challenge of grave impact in the real-world. In this paper, we outline the mathematical model of the network security domain. We provide the mathematical formulation for the two person security game between the defender and the attacker, where the attacker sends malicious packets from some (known) set of sources and the defender uses packet inspections to detect such malicious traffic. Since the nonlinear calculations render the model intractable for computer networks with even 5 sources and 5 targets, we also introduce a zero-sum simplification of the original model. We then propose GRANDE, a novel error-bounded approximation algorithm that relies on the submodular property of the malicious packet detection problem.We validate our algo-

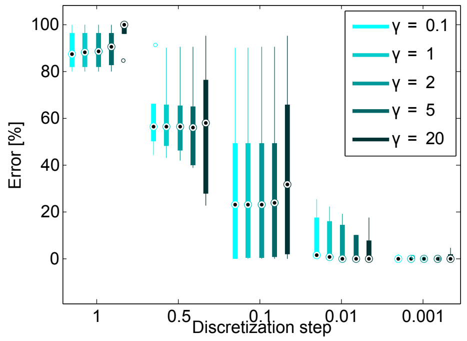


Figure 4: Error of Grande evaluated over different scale-free graphs. The errors are grouped according d, and in each group, γ was set to $\{0.1, 1, 2, 5, 20\}$.

rithms experimentally, and show that Grande, on average, produces results with orders of magnitude higher solution quality as projected by the theoretical worst case bounds. This work contributes by outlining the challenges present in the network security domain, and by introducing state-of-the-art algorithms that compute the optimal strategy for the defender in computer networks.

7. ACKNOWLEDGMENTS

This research was supported by the United States Department of Homeland Security through the National Center for Risk and Economic Analysis of Terrorism Events (CREATE) under award number 2010-ST-061-RE0001, by the Czech Ministry of Education, Youth and Sports (grant no. LH11051), by the Czech Science Foundation (grant no. P202/12/2054) and by the AirForce Office of Scientific Research, Air Force Material Command, USA (grant no. FA8655-10-1-3016).

8. REFERENCES

[1] T. Alpcan. *Network Security: A Decision and Game-Theoretic Approach*. Cambridge University Press, 2010.

[2] L. Chen and J. Leneutre. A game theoretical framework on intrusion detection in heterogeneous networks. *IEEE Transactions on Information Forensics and Security*, 4(2):165–178, 2009.

[3] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 82–90. ACM, 2006.

[4] J. Czyzyk, M. Mesnier, and J. Moré. The NEOS server. *Computational Science & Engineering, IEEE*, 5(3):68–75, 1998.

[5] S. Dughmi. Submodular Functions: Extensions, Distributions, and Algorithms. A Survey. *CoRR*, 2009.

[6] Y. Freund and R. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

[7] E. Halvorson, V. Conitzer, and R. Parr. Multi-step Multi-sensor Hider-Seeker Games. *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[8] L. Heberlein and M. Bishop. Attack class: Address spoofing. In *Proceedings of the 19th National Information Systems Security Conference*, pages 371–377, 1996.

[9] M. Jain, D. Korzhyk, O. Vaněk, V. Conitzer, M. Pěchouček, and M. Tambe. A double oracle algorithm for zero-sum security games on graphs. In *Proceedings of the Tenth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Taipei, Taiwan*, 2011.

[10] M. Jain, J. Tsai, J. Pita, C. Kiekintveld, S. Rathi, M. Tambe, and F. Ordóñez. Software Assistants for Randomized Patrol Planning for the LAX Airport Police and the Federal Air Marshals Service. *Interfaces*, 40:267–290, 2010.

[11] M. Kodialam and T. Lakshman. Detecting network intrusions via sampling: a game theoretic approach. In *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. (INFOCOM)*, volume 3, pages 1880–1889. IEEE, 2003.

[12] D. Korzhyk, V. Conitzer, and R. Parr. Security games with multiple attacker resources. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

[13] A. Krause and C. Guestrin. Near-optimal observation selection using submodular functions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2007.

[14] A. Krause, A. Roper, and D. Golovin. Randomized sensing in adversarial environments. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2133–2139, 2011.

[15] M. Manshaei, Q. Zhu, T. Alpcan, T. Basar, and J. Hubaux. Game theory meets network security and privacy. *EPFL, Lausanne, Tech. Rep*, 2010.

[16] G. Nemhauser and L. Wolsey. Maximizing submodular set functions: formulations and analysis of algorithms. *Studies on Graphs and Discrete Programming*, pages 279–301, 1981.

[17] H. Otrok, M. Mehrandish, C. Assi, M. Debbabi, and P. Bhattacharya. Game theoretic models for detecting network intrusions. *Computer Communications*, 31(10):1934–1944, 2008.

[18] P. Paruchuri, J. Pearce, J. Marecki, M. Tambe, F. Ordonez, and S. Kraus. Playing games for security: an efficient exact algorithm for solving Bayesian Stackelberg games. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 895–902, 2008.

[19] V. Paxson. Bro: A system for detecting network intruders in real-time. *Computer networks*, 31(23-24):2435–2463, 1999.

[20] J. Pita, C. Kiekintveld, M. Tambe, E. Steigerwald, and S. Cullen. GUARDS - Game Theoretic Security Allocation on a National Scale. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2011.

[21] L. Schrage and I. LINDO Systems. Optimization modeling with lingo. 1999.

[22] J. Vondrak. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, pages 67–74, 2008.

Sustaining Cooperation on Networks: An Analytical Study based on Evolutionary Game Theory

Raghunandan M. A.
Dept. of Computer Science and Automation,
Indian Institute of Science, Bangalore, India
raghunandan@csa.iisc.ernet.in

Subramanian C. A.
Orca Radio Systems,
Bangalore, India
subbu@orcasystems.com

ABSTRACT

We analytically study the role played by the network topology in sustaining cooperation in a society of myopic agents in an evolutionary setting. In our model, each agent plays the Prisoner's Dilemma (PD) game with its neighbors, as specified by a network. Cooperation is the incumbent strategy, whereas defectors are the mutants. Starting with a population of cooperators, some agents are switched to defection. The agents then play the PD game with their neighbors and compute their fitness. After this, an evolutionary rule, or imitation dynamic is used to update the agent strategy. A defector switches back to cooperation if it has a cooperator neighbor with higher fitness. The network is said to sustain cooperation if almost all defectors switch to cooperation. Earlier work on the sustenance of cooperation has largely consisted of simulation studies, and we seek to complement this body of work by providing analytical insight for the same.

We find that in order to sustain cooperation, a network should satisfy some properties such as small average diameter, densification, and irregularity. Real-world networks have been empirically shown to exhibit these properties, and are thus candidates for the sustenance of cooperation. We also analyze some specific graphs to determine whether or not they sustain cooperation. In particular, we find that scale-free graphs belonging to a certain family sustain cooperation, whereas Erdos-Renyi random graphs do not. To the best of our knowledge, ours is the first analytical attempt to determine which networks sustain cooperation in a population of myopic agents in an evolutionary setting.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Multiagent systems

General Terms

Economics, Theory

Keywords

Agent Interaction, Evolution of Cooperation, Emergent behavior

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

1. INTRODUCTION

The question of how cooperation emerges among rational, intelligent agents is one which has received considerable amount of attention. When agents interact with one another, they are often faced with two choices - cooperate with each other for mutual benefit, or think about one's own interests, and defect. This abstract interaction model is captured succinctly in the Prisoner's Dilemma (PD) game, which is a two player, one shot, simultaneous move game. Although the Hawk-Dove and Stag Hunt games are also used to model agent cooperation, the PD game is arguably the most often used and well studied.

Table 1 shows the (normalized) payoff matrix for the two player PD game. In this game, each agent can choose one of two actions or strategies – cooperate or defect. The first entry in each cell in the table is the payoff for the row player, and the second entry is that of the column player. For example, when the row player cooperates and the column player defects, then the cell entry is $0, b$, meaning that the row player gets 0, and the column player gets b. When both agents cooperate, each gets a moderate payoff (say 1). However, a defector achieves a higher payoff (say b) against a cooperator, who gets zero payoff. Here b is called the benefit or temptation to defect, and typically $1 < b < 2$. When both agents defect, then both get zero payoff. We observe that if the column player cooperates, the row player is better off defecting, whereas, if the column player defects, then the row player is indifferent as to his strategy. Hence defection is a dominant strategy, and rational agents will be expected to defect always. When two rational players play the PD game, both defect. However, it would ultimately have been better for both agents to have cooperated with each other (mutual cooperation is *Pareto-optimal* with respect to mutual defection) and therein lies the dilemma. Hence it is of interest to study what conditions or protocols of interaction induce agents to cooperate with one another.

	Cooperate	Defect
Cooperate	1,1	0,b
Defect	b,0	0,0

Table 1: Prisoner's Dilemma payoff matrix

In the literature, several settings that sustain cooperation have been proposed and studied. When agents interact repeatedly with one another, and can remember past histories, then agents can retaliate against defectors by refusing to cooperate in future interactions, and this could induce mutual cooperation [3]. However this line of reasoning fails in settings where the same agents may not interact repeatedly. In such settings, reputation mechanisms may be used by the community of agents to track and punish defectors [19]. Such a technique requires unique, constant identities for the defectors, and cannot account for settings in which agents are anonymous, or can freely change their identities, such as happens in interactions over the Internet. Hence we seek interaction models which sustain cooperation in a society of myopic, memoryless agents. We approach this study by looking at the way in which the structure of interaction between the agents affects their strategies.

In a multiagent society, one cannot always expect that all agents interact with one another. For example, spatial structure (agents only interact with other agents in their vicinity) and organizational structure (agents interact only with other agents immediately above or below them in a hierarchy) are two immediate examples in which interactions are restricted between agents. The structure of the interaction of agents with one another is naturally represented by a network, with each node corresponding to an agent, and each edge representing an interaction between a pair of agents. We model agent interaction as an evolutionary PD game played on this network. In this model, every agent is either a cooperator or a defector, and plays the same strategy uniformly with each of its neighbors on the network. We take cooperation to be the incumbent strategy, and defection as the mutant strategy. Starting with a population of cooperators, a small number of agents are switched to defection. The agents then play the game with their neighbors on the network. The fitness of an agent is calculated as the sum of the payoffs that it receives in each game. After one such round the defectors then decide whether to stay with their current strategy, or switch, depending on the fitness of their neighbors. In particular, a defector switches to cooperation when one of its neighbors is a cooperator with higher fitness. We say that a network sustains cooperation if almost all defectors switch to cooperation.

Given this simple model of agent interaction, we ask which networks sustain cooperative behavior, and which do not. In our analysis, we identify some necessary properties that a network should satisfy in order to sustain cooperation, such as small average diameter, densification, and irregularity (Section 4). Real-world networks have been empirically observed to exhibit these properties, and hence can be considered suitable for sustaining cooperation. We also identify some graphs which sustain cooperation, and some which do not (Section 5). In this way, we try to build a complete characterization of networks which sustain cooperation.

There have been many simulation studies on the sustenance of cooperation on networks, but very few of analytical nature. In an analytical study, although the complexity of the model that is studied is necessarily limited in the interest of tractability, the insights obtained are very clear. In fact, it is the analytical approach alone that enables us to obtain a characterization of graphs that sustain cooperation, as illustrated in our necessary conditions above. In contrast, it is arguably difficult, if not impossible to conduct an simulation on a graph class such as “all graphs with large average diameter”.

The rest of the paper is as follows: In Section 2, we describe the network interaction model that we use in our study, and formally define when a network is said to sustain cooperation. In Section 3, we briefly survey the literature relating to the study of sustenance of cooperation under various settings. In Sections 4 and 5, we give proofs for the necessary conditions to sustain cooperation and analyze some specific graphs, respectively. Finally, in Section 6, we summarize our study, and also identify some avenues for future work.

2. THE NETWORK INTERACTION MODEL

2.1 Intuition for the Model

Our definition of networks which sustain cooperation uses a fitness function and imitation rule that follows in spirit the model described by Kearns and Suri [13]. However, we have made some important changes to the model to make it more realistic and suitable to a wider range of applications. We now describe the salient features of our model.

1. **Agent fitness**: The fitness of each agent is defined to be the sum of the payoffs it derives from the PD games played with each of its neighbors in the interaction network. We have defined it to be the sum rather than average (as considered in [13]), because in real world networks, the agents are disparate and have fitnesses based on their centrality and degree of connectedness and normalizing them based on their degree is not natural. The sum of payoffs has been used in earlier works as a measure of fitness of an agent in evolutionary game theory [1, 17].

2. **Incumbents and mutants**: In [13], it is shown that an evolutionarily stable strategy (ESS) in a classical sense (that is, when underlying network of interactions is a complete graph) is also an ESS on graphs (with respect to their fitness model). This result holds under mild restrictions on the graph or how the mutants are chosen. In the context of the PD game, this implies that a population of defectors is resistant to invasion by mutant cooperators. However, we are trying to study whether cooperation (a dominated strategy) can survive mutant attacks (by defectors) given certain network topologies. To this effect we assume the incumbent strategy is cooperation and introduce defectors as mutants.

3. **Imitation dynamics**: In a PD game, for every agent, the best response strategy is to defect (irrespective of the actions of other agents). Hence cooperation cannot be sustained when agents follow best response dynamics. The dynamics considered in our work is not that of best response but that of imitation (which not only applies to humans but also to primitive life forms where rationality cannot be completely justified). Here, if a mutant defector has at least one cooperator neighbor with a higher fitness than it, then it is likely to imitate that strategy and switch over to cooperation. This is a natural behavior and models the fact that every agent tries to imitate other successful agents it interacts with, so as to improve its own payoff [8].

4. **Sustaining cooperation**: Starting with a network of cooperators, a small randomly selected fraction of agents are switched to defection. We say that a network sustains cooperation, if almost all defectors change

their strategy back to cooperation, in accordance with the imitation dynamics described above. The random selection is warranted, as with adversarial placement it is always possible to place mutants in such a way that no mutant switches to cooperation, and the question becomes trivial.

2.2 Notation

Following the game model considered by Santos and Pacheco [20], we work with a normalized PD game matrix, shown in Table 1. The interaction structure among the agents is specified by a graph $G = (V, E)$, where V is the vertex (or node) set, and $E \subseteq V \times V$ is the edge set. The number of vertices (|V|) is denoted as n. We consider graphs which are specified for all large values of n, as we are interested in asymptotic behavior as $n \to \infty$. When two vertices u, v are connected by an edge, we denote the edge as (u, v) (which is equivalent to (v, u), because we work with undirected graphs), and say that $(u, v) \in E$. For each vertex $v \in V$, the neighborhood $N(v)$ is the set of vertices adjacent to v. That is, $N(v) = \{u \in V : (u, v) \in E\}$. Each vertex v corresponds to an agent, and has a fixed strategy $s(v)$, which is either C or D. In future, we will refer to the terms agent and vertex interchangeably. Also, we will refer to each agent, or node, as either a cooperator or a defector, depending on its strategy. Now each agent plays the PD game with each of its neighbors on the graph. Denote by $f(v, u)$, the payoff obtained by agent v playing against agent u. Now the total fitness $f(v)$ of agent v is the sum of its payoffs against each of its neighbors, that is, $f(v) = \sum_{u \in N(v)} f(v, u)$.

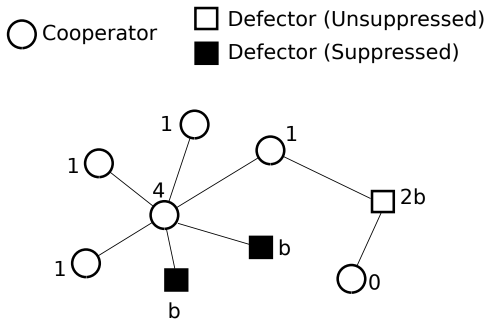


Figure 1: Network Interaction Model

Figure 1 shows an example graph with some cooperators and defectors, along with their fitness. We also see two defectors (shaded) adjacent to a cooperator who has a higher fitness than them. We say that such defectors are *suppressed*. If a defector is not adjacent to any cooperator of higher fitness than it, then we refer to it as *unsuppressed*. We say that an event occurs with high probability, if it occurs with probability $1 - o(1)$ (where $o(1)$ is a term which goes to 0 as $n \to \infty$). Given a graph G, and a constant ϵ^*, $(0 \leq \epsilon^* \leq 1)$ the defector selection process chooses a random subset of vertices D, of size $\epsilon^* n$ as defectors, keeping the other $(1 - \epsilon^*)n$ vertices C as cooperators. (Strictly speaking, the number of cooperators and defectors should be integers. Our analysis can be carried out taking either the floor or ceiling of such quantities, without affecting the results.) A graph G is said to sustain cooperation, if there is a constant ϵ $(0 < \epsilon < 1)$ such that for all values $\epsilon^* \leq \epsilon$, with high probability (with respect to the defector selection process), at most $o(n)$ defectors are unsuppressed. If not, that is, for all values of ϵ, there exist corresponding values of $\epsilon^* \leq \epsilon$, such that the probability that the number of unsuppressed defectors is $\Omega(n)$ is greater than some fixed α, then we say that the graph does not sustain cooperation. The definition of sustenance of cooperation of a graph, can be extended to that of a random graph in a simple manner. In this case, we require that with high probability with respect to selection of a graph from the set of possible graphs (in addition to selection of the defector set), at most $o(n)$ defectors are unsuppressed.

In the above model, we select a random subset of nodes as defectors. One might also consider an adversarial selection. In Section 4.1, we show that with adversarial selection of the defector set, no graph can sustain cooperation, and hence the model becomes uninteresting.

3. RELATED WORK

We now briefly survey the literature which addresses the broad question of how cooperation emerges and is sustained in social interactions, and also identify some key differences between earlier work and ours. These studies can be categorized according to the choice of conditions studied. The most commonly used and popular model of agent interaction is the Prisoner's Dilemma (PD) game, although some studies use the Hawk-Dove (HD, also called Snowdrift or Chicken) game [10, 20]. The network of interaction between the agents can be either static [11, 20] or dynamic [9, 12]. The agents themselves can be myopic and memoryless [11, 20] or strategic and intelligent [3, 12].

Early studies relating to sustenance of cooperation have been in the context of repeated games, where agents play the PD game repeatedly with one another for an infinite number of rounds. In such a setting, the Folk Theorem indicates that mutual cooperation can be sustained as a Nash equilibrium [18]. Axelrod and Hamilton [3] found experimentally that cooperation is sustained when defectors are punished reciprocally with defection in a tit-for-tat fashion. The study was conducted in the form of two computer tournaments where strategic programs competed against one another in a repeated PD game. In their model, each agent (program) plays with every other agent in several rounds, can distinguish each agent's identity, remember the history of actions for each agent that it played, and adopt a different strategy for each interaction. Also the final fitness of an agent is calculated as the sum of the payoffs that it received in each round. A detailed analysis of the sustenance of cooperation in such a setting can be found in [2]. In contrast to this model, agents in our model can adopt only one strategy at a time, which they uniformly exercise in all their interactions, and the payoffs that agents receive are recomputed in each round and are not added up.

Hofmann, Chakraborty, and Sycara [11] carried out a comprehensive simulation study, in which they found that the sustenance of cooperation depends on a number of factors such as network topology, strategy update rule, and initial population of cooperators. In particular, they found that scale-free networks sustain cooperation given almost any update rule. Hanaki, Peterhansl, Dodds, and Watts [9] conducted simulations and found that cooperation can be sustained in a dynamic multiagent network when links between agents are costly and local structure is largely absent. They used a model where each agent imitates the strategy of its most successful neighbor, and breaks/creates links stochastically based on a cost/benefit comparison. Santos and Pacheco [20] carried out simulations of the PD and HD games on various networks and found that cooperation is not sustained on regular graphs and graphs formed by a growth model without preferential attachment, but can be sustained on graphs formed by a growth model with preferential attachment (such graphs were introduced by Barabasi and Albert [4]). Other simulation studies of the sustenance of cooperation are [1, 6, 10, 15, 17, 21]. Also refer to Szabo and Fath [22] for a survey of evolutionary games on graphs.

We observe that the literature in this field largely deals with simulation studies of the sustenance of cooperation on networks. We now turn to the relevant analytical studies in this area. The work which is most similar to ours in terms of the techniques used is that of Kearns and Suri [13], who extended the notion of evolutionarily stable strategies (ESS, [16]) to games played on graphs. An ESS is a strategy that is resistant to invasion by mutant strategies. Kearns and Suri showed analytically that the ESS of games are preserved in their model. In the context of the PD game, this implies that defection is dominant even in the graph setting. While the techniques that we use are similar to the above work, our model is fundamentally different in the computation of fitness, as well as in the restrictions imposed on the graph and placement of defectors. Hence our analysis yields qualitatively different results regarding the sustenance of cooperation, which cannot be obtained in their model. Immorlica, Lucier, and Rogers [12] found that cooperation can be sustained by the formation of social capital. They studied a PD game on a dynamic network where an agent can change its links, but not its strategy. At each round some randomly chosen agents are removed, and replaced by new agents, who choose their strategy based on expected long term fitness. They found that under some parameter settings, cooperators and defectors co-exist in a dynamic self-correcting equilibrium. Our work is different from this, in that we consider myopic, memoryless agents, who cannot compute long-term costs and benefits, but are instead driven by imitation dynamics.

3.1 Our Contributions

In the context of the proposed model, we identify the following necessary conditions for a network to sustain cooperation:

- **Small Average Diameter**: The average diameter of the network should be sub-linear in the number of nodes. Real-world networks have been shown to have an average diameter that grows logarithmically in the number of nodes [23].

- **Densification**: As the number of nodes in the network grows, the average degree of the nodes should increase. In other words, the number of edges should grow super-linearly in the number of nodes [14].

- **Irregularity**: The ratio of maximum degree to minimum degree should be greater than b, the benefit received by defectors. Real-world networks have a power law degree distribution, and hence satisfy this condition [4].

In particular, we analyze the sustainability of cooperation on specific important networks, and classify them accordingly:

- **Sustaining cooperation**: Scale-Free graphs, Hierarchical graphs, Bipartite Random graphs

- **Not sustaining cooperation**: Erdos-Renyi random graphs

4. NECESSARY CONDITIONS FOR SUSTAINING COOPERATION

First, we show that with adversarial selection of the defector set, no graph can sustain cooperation.

4.1 Adversarial Mutant Selection

THEOREM 1. *For any graph G with n nodes, it is always possible to place ϵn defectors and $(1-\epsilon)n$ cooperators in such a way that for every cooperator v_c and defector v_d adjacent to each other, $f(v_d) \geq f(v_c)$.*

PROOF. Let P be a placement of cooperators and defectors on the graph which minimizes the total fitness of all cooperators. We will show that this placement satisfies the condition of the theorem, and we are done.

If not, then there is a cooperator v_c, and a defector v_d adjacent to each other, such that $f(v_c) > f(v_d)$. Let the number of cooperators who are adjacent to v_c but not v_d be k_c and those adjacent to v_d but not v_c be k_d, and let k be the number of cooperators adjacent to both. Then, $f(v_c) = k_c + k; f(v_d) = b \cdot (k_d + k); f(v_c) > f(v_d) \Rightarrow k_c + k > b \cdot (k_d + k) \Rightarrow k_c > k_d$. Now interchange the strategies of v_c and v_d. That is, v_c now becomes a defector, and v_d becomes a cooperator. Let us consider the total change in the cooperator fitness. The fitness of each of the k_c cooperators adjacent to v_c decreases by 1 $(-k_c)$; that of the k_d cooperators adjacent to v_d increases by 1 $(+k_d)$; and that of the k cooperators adjacent to both does not change. Also, v_c is no longer a cooperator $(-k_c - k)$ and v_d is now a cooperator $(+k_d + k)$. Hence the change in total cooperator fitness is $2(k_d - k_c)$, which is negative since $k_d < k_c$. Now we have a new placement of cooperators and defectors P', in which the total cooperator fitness is strictly less than that of P, contradicting the minimality of P. $\square$

We now establish two key lemmas that will be used in the proofs.

LEMMA 1. *Let A_1, A_2, ..., A_k be a set of events, where k is a constant. Further, each A_i occurs with high probability Let $A = \bigcap_{i=1}^{k} A_i$. Then, A occurs with high probability.*

PROOF. For each i, since $Pr[A_i] = 1 - o(1)$, $Pr[\overline{A_i}] = o(1)$. Also, $Pr[A] = 1 - Pr[\overline{A}]$, and $\overline{A} = \bigcup_{i=1}^{k}(\overline{A_i})$. By the union bound, $Pr[\bigcup_{i=1}^{k}(\overline{A_i})] \leq \sum Pr[\overline{A_i}] = o(1)$. $\square$

LEMMA 2. *Let a graph G have $\Omega(n)$ vertices of bounded degree. Then G does not sustain cooperation.*

PROOF. Let V_S be the set of vertices of bounded degree, that is, which have degree at most some k. We are given that for some fixed β, $|V_S| \geq \beta n$. (Actually, this statement is true when $n > n_0$, for some fixed n_0, but we are interested in the asymptotic behavior as $n \to \infty$, so we do not mention this condition.) Set $\epsilon^* = \epsilon$. For each vertex in this set, the probability that it and its neighbors are defectors is at least ϵ^{k+1}, and expected number of such vertices is at least $\beta\epsilon^{k+1}n$, which is linear in n. It is clear that these vertices are unsuppressed, and hence the total number of unsuppressed defectors is at least $\beta\epsilon^{k+1}n$. By the Large Expectation Lemma (refer A.2), we can say that with constant non-zero probability, the number of unsuppressed defectors is greater than $o(n)$. $\square$

We now establish the necessary conditions for any graph to sustain cooperation.

4.2 Small Average Diameter

With a view to keeping our model as general as possible, we do not require the networks under consideration to be connected. That is, the network may consist of disjoint sets of vertices, which are not connected to each other. Each such set of vertices, within which there is a path between any pair of vertices is called a component. We now define the diameter and average diameter of a component G' or a network G. The distance between two nodes in a component is defined as the length of the shortest path between them. The diameter ($diam(G')$) and average diameter ($diam_{avg}(G')$) of a component G' are defined as the maximum and average distance respectively over all pairs of nodes within the component. The diameter ($diam(G)$) and the average diameter $diam_{avg}(G)$ is defined as the maximum over all components of the diameter and average diameter respectively.

We find that in order to sustain cooperation, the average diameter of the network should be $o(n)$, that is, the average diameter should grow sub-linearly in the number of nodes. In other words, the network cannot have a large average diameter of $\Omega(n)$.

THEOREM 2. *Let G be a graph of large average diameter, that is, $diam_{avg}(G) = \Omega(n)$. Then G does not sustain cooperation.*

PROOF. From the definition for average diameter, we know that some component of G, say G', has average diameter $diam_{avg}(G') = \Omega(n)$. It is easy to see that this implies $diam(G') = \Omega(n)$ and also that G' has $\Omega(n)$ vertices. We now show that some $\Omega(n)$ vertices in G' have bounded degree. This along with Lemma 2 establishes the theorem.

Suppose not, that is, at most $o(n)$ vertices in G' have bounded degree. Call this set V_S. All the other vertices in G' have degree $\omega(1)$. Call this set V_B. Recursively apply the following procedure:

1. Let $i = 0$; $V' = V_B$
2. Find $v \in V' : N(v) \bigcap (S_0 \biguplus S_1 .... \biguplus S_{i-1}) = \phi$ (Stop if no such vertex exists).
3. Let $N[v] = v \cup N(v)$
4. Assign $S_i = N[v]$; $i = i + 1$
5. Assign $V' = V' - N[v]$

Let the value of i at the end of the iterations be k. Since each S_i has $\omega(1)$ vertices, the number k of stars S_i formed is $o(n)$. Notice that each vertex remaining in V' at the end is at distance one to some star, that is, it has a neighbor in some star S_i. For each vertex v remaining in V', find some i such that $S_i \cap N(v) \neq \phi$, and add v to S_i. Now the diameter of each S_i is at most 4.

The component G' is now decomposed into two parts - $o(n)$ stars $(S_0, ..., S_{k-1})$, and some subset V'_S of V_S. Let us look at the shortest path between any two vertices in G'. This path passes through each S_i at most constant times, and through each of the V'_S at most once. Hence the diameter is $o(n)$, contradicting the statement above that $diam(G') = \Omega(n)$. $\square$

4.3 Densification

A graph is said to densify if the number of edges in the graph asymptotically grows faster than n. That is, $|E| = \omega(n)$. Put in another way, these are graphs whose average degree increases with n. This rules out all sparse graphs (paths, cycles, trees, planar graphs, etc.), that is, graphs which do not densify over time.

THEOREM 3. *Let G be a sparse graph, that is, $|E| = O(n)$. Then G does not sustain cooperation.*

PROOF. We will show that some $\Omega(n)$ vertices have bounded degree, which along with Lemma 2, establishes the result.

We are given that for some fixed β, $|E| \leq \beta n$. Since the total degree is at most $2\beta n$, there can be at most $n/2$ vertices of degree greater than 4β, and hence at least $n/2$ vertices of degree bounded by 4β, which is a constant. $\square$

4.4 Irregularity

The degree $d(v)$ of a vertex v is the size of its neighborhood $N(v)$. The maximum degree Δ and minimum degree δ of a graph G are defined as the maximum and minimum respectively, over the degrees of all vertices of G. For a graph G to sustain cooperation, the ratio of the maximum degree to the minimum degree should be at least b ($\Delta/\delta \geq b$). This means that the graph should be irregular to some extent, ruling out all near-regular graphs.

THEOREM 4. *Let G be a Near-Regular graph, that is, $\Delta/\delta < b$. Then G does not sustain cooperation.*

PROOF. Let $\tau := \Delta/\delta < b$. If $\delta = O(1)$, that is, a constant, then Δ is also a constant, implying that all nodes have bounded degree. By Lemma 2, the graph does not sustain cooperation.

Now consider $\delta = \omega(1)$. We will show that with fixed non-zero probability, there is a linear-sized set of defectors, in which each defector has a fitness higher than that of any other cooperator. No defector in this set is suppressed, and the result follows.

The maximum degree of any vertex is Δ, and hence the maximum fitness of any cooperator is Δ. This implies that if a defector has more than $\frac{\Delta}{b}$ cooperator neighbors, then it is unsuppressed. We call such a node *bad*, and the other nodes *good*. Let us now give an upper bound for the probability of a node being good. This happens if the node is a cooperator (probability $1 - \epsilon^*$), or if the node has less than $\frac{\Delta}{b}$ cooperator neighbors. Using the Chernoff bound, this latter probability is not more than $e^{-\frac{\lambda^2}{3}\mu} =: \beta$, where μ is the expected number of cooperator neighbors of a vertex, which is at least $\delta(1-\epsilon^*)$, and $\lambda = (1 - \frac{\tau}{(1-\epsilon^*)\cdot b})$ is the deviation from the mean on the lower side (which is strictly positive, as ϵ^* can be set to a value less than $1-\frac{\tau}{b}$). Using the Union bound, the probability of a node being *good* is not more than $1-\epsilon^*+\beta$, and hence the probability of a node being *bad* is at least $\epsilon^*-\beta =: \gamma$, which is strictly positive (as $\delta = \omega(1)$, β is arbitrarily close to zero).

The expected number of *bad* nodes is at least γn. The maximum number of bad nodes is $\epsilon^* n$, and hence by the Large Expectation Lemma (refer A.2), the size of this set is linear with non-zero probability. $\square$

5. SUSTAINABILITY OF COOPERATION ON SPECIFIC GRAPHS

5.1 Erdos-Renyi Random Graphs

We now analyze the $G(n,p)$ model of Erdos-Renyi [7]. In this model, a graph on n nodes is taken, and the edge between every pair of vertices is included in the graph independently with probability $p(n) = p$. In such a graph, the degree of each vertex is roughly close to the expected value (np). Hence this graph behaves like a regular graph, and one would expect that does not sustain cooperation. This intuition is supported by the proofs for specific ranges of the parameter p ($p = O\left(\frac{1}{n}\right)$ and $p = \omega\left(\frac{log(n)}{n}\right)$). We expect the result to hold similarly for other values of p as well.

THEOREM 5. *Let $G(n,p)$ be an Erdos-Renyi random graph, where $\forall e \in V \times V, Pr[e \in E] = p$, and $p = O\left(\frac{1}{n}\right)$ or $p = \omega\left(\frac{log(n)}{n}\right)$. $G(n,p)$ does not sustain cooperation.*

PROOF. When $p = O\left(\frac{1}{n}\right)$, with high probability the number of edges is linear in n. That is, $|E| = O(n)$. Hence $G(n,p)$ does not densify, and by Theorem 3, does not sustain cooperation.

When $p = \omega\left(\frac{log(n)}{n}\right)$, we show that the graph is near-regular, and hence by Theorem 4, does not sustain cooperation. The expected degree of each vertex is $\mu := E[d(v)] = (n-1)p = \omega(log(n))$. By the Chernoff bound, the probability that the degree of a node is not within $(1 \pm \lambda)\mu$ is at most $2e^{-\frac{\lambda^2}{3}\mu}$, and the expected number of such nodes is $n \cdot 2e^{-\frac{\lambda^2}{3}\mu}$, which goes to 0. Hence all nodes have degrees within $(1 \pm \lambda)$ of the expected value. Setting λ such that $\frac{1+\lambda}{1-\lambda} < b$ makes the graph near-regular, and we have the result. $\square$

5.2 Bipartite Random Graphs

These graphs are bipartite graphs, that is, graphs in which the vertex set can be partitioned into two sets L and R, such that there are no edges within each partition. Also the size of the partition L is given by a function $f(n)$, which asymptotically grows faster than $log(n)$, but slower than n. Every edge between one vertex in L and one in R is included in the graph independently with probability $p(n)$ (which is at least $\frac{4log(n)}{f(n)}$).

We thus have a family of random graphs parametrized by the values of $f(n)$ and $p(n)$. We observe that the expected degrees of vertices in partition L is linear in n, whereas in partition R is sub-linear in n, because of which, cooperators in L can suppress defectors in R, and intuitively these graphs should sustain cooperation. This family of graphs is useful in that allows us to construct networks which sustain cooperation having any given edge density which is super-linear in $n * log(n)$ and sub-linear in n^2, that is $|E| = \omega(n * log(n)), o(n^2)$. This is done by setting $f(n) = \frac{|E|}{n}$, and $p(n)$ appropriately.

THEOREM 6. *Let G be a bipartite graph with $V = L \biguplus R$ (L and R are the two partitions of the vertices), $|L| = f(n) = \omega(log(n)), o(n)$, and $\forall e \in L \times R, Pr[e \in E] = p(n) = p$, where $p(n) \geq \frac{4log(n)}{f(n)}$. G sustains cooperation.*

PROOF. We will show that all defectors in R are suppressed. Hence the number of unsuppressed defectors is only that in L, which is at most $f(n) = o(n)$, and the theorem is proved.

Let $0 < \epsilon < 1$ and let ϵ^* take any value $\leq \epsilon$; define $\epsilon' := 1 - \epsilon^*$. The number of cooperators in partition L has an expected value of $\epsilon' f(n)$, and by the Chernoff bound, is at least $\frac{1}{2}\epsilon' f(n)$ with high probability. Call this set V_C.

There are a total of $\epsilon' n$ cooperators in the graph, and hence at least $\epsilon' n - f(n)$ cooperators in R. Since $f(n) = o(n)$, this number is at least $\frac{1}{2}\epsilon' n$. Since a cooperator in V_C is adjacent to each cooperator in R independently with probability p, the expected fitness of a cooperator in V_C is at least $\frac{1}{2}\epsilon' np$. By the Chernoff bound, the probability that the fitness is less than half of this expected value is not more than $e^{-\frac{1}{24}\epsilon' np}$. The expected number of such vertices is not more than $\frac{1}{2}\epsilon' n * e^{-\frac{1}{24}\epsilon' np} = \frac{1}{2}\epsilon' e^{log(n) - \frac{1}{24}\epsilon' np}$, which goes to 0, as $p \geq \frac{48log(n)}{n}$. Hence the fitness of each cooperator in V_C is at least $\frac{1}{4}\epsilon' np$, with high probability.

Using a similar analysis, we can say that the fitness of each defector $d \in R$ is at most $\frac{3}{2}b\epsilon' pf(n)$, with high probability.

From the above two arguments, we can say that any defector in R who is adjacent to any one cooperator in V_C is suppressed.

The probability that a defector in R is not connected to any cooperator in V_C is at most $(1-p)^{\frac{1}{2}\epsilon' f(n)}$, which is at most $e^{-\frac{1}{2}\epsilon' f(n)p}$. The expected number of such defectors is at most $n * e^{-\frac{1}{2}\epsilon' f(n)p}$, which goes to 0 since $p \geq \frac{4log(n)}{f(n)}$. $\square$

5.3 Hierarchical Graph

This graph is built by starting with a complete binary tree, and connecting each node to all of its ancestors (and descendants). The set of nodes of a given degree correspond to a *level* in the hierarchy. Nodes which are higher up in the hierarchy are high degree nodes, and those lower in the hierarchy are of low degree. Hence intuitively cooperators in higher levels can suppress defectors in lower levels, and cooperation can be sustained.

THEOREM 7. *Let G be a complete binary tree where each node is connected to all its descendants. Formally, let Σ^* be the set of all strings over alphabet $\Sigma = \{0,1\}$. For each such $s \in \Sigma^*$, let $|s|$ denote the length of s, and let prefix be a relation on Σ^*, such that prefix(u,v) is true, when the string u is a prefix of the string v. Take $V = \{s \in \Sigma^* : |s| \leq log(n+1) - 1\}$, and $E = \{(u,v)$: prefix(u,v) or prefix$(v,u)\}$. G sustains cooperation.*

PROOF. Let us call the length of the string that represents a vertex v, as the *level* of v; and say that the level of vertex v is *above*, or *higher* than that of vertex u, if $|v| < |u|$ (and also that the level of vertex $|u|$ is *below* or *lower* than that of $|v|$). Let $0 < \epsilon < 1/2$ and let ϵ^* take any value $\leq \epsilon$, and define $\epsilon' := 1 - \epsilon^*$.

First we will show that every vertex at level $h_1 + 1$, where $h_1 = \frac{1}{3}log(n)$, has an ancestor who is a cooperator (and hence every vertex below level h_1 (that is, $|v| > h_1$) has an ancestor of level higher than $h_1 + 1$ (that is, $|v| \leq h_1$), who is a cooperator). Every vertex at level $h_1 + 1$ has h_1 ancestors, and the probability that all of them are defectors is roughly $(\epsilon^*)^{h_1}$. Hence the expected number of such vertices is $2^{h_1+1} \cdot (\epsilon^*)^{h_1}$, which goes to 0 as $\epsilon^* \leq \epsilon < 1/2$.

Now, we will show that each vertex of level h_1 (and hence each vertex of level higher than $h_1 + 1$), has at least $\frac{1}{4}\epsilon' n^{2/3}$ cooperator neighbors. For each vertex at level h_1, the expected number of cooperator neighbors is at least $\frac{1}{2}\epsilon' n^{2/3}$, and by the Chernoff bound, the probability that this number is less than half the expected value is not more than $e^{-\frac{1}{12}\epsilon' n^{2/3}}$. The expected number of such vertices is not more than $n^{2/3} * e^{-\frac{1}{12}\epsilon' n^{2/3}}$, which goes to zero.

It is clear that each defector of level lower than $h_2 = \frac{2}{3}log(n)$ has fitness f_d at most $b \cdot |N(d)| \leq b \cdot (n^{1/3} + log(n))$. By the above two arguments, each such defector has an ancestor of level higher than $h_1 + 1$ who is a cooperator, who has fitness f_c at least $\frac{1}{4}\epsilon' n^{2/3}$. Clearly $f_c > f_d$. Hence each such defector is suppressed. The number of unsuppressed defectors is at most that in levels above $h_2 + 1$, which is $O(n^{2/3}) = o(n)$. $\square$

5.4 Scale-Free Graphs

In order to study the sustainability of cooperator on scale-free networks, we consider the random scale-free graph described in [5]. In this model, the vertices are labeled from 1 through n, and the edge between two vertices is included in the graph with some probability, which is defined as a function of the vertex labels. In our model, $Pr[(i,j) \in E] = (ij)^{-\frac{1}{2}+\kappa}$, where $0 < \kappa < \frac{1}{2}$. The expected degree of node i is given by $E[d(i)] = \frac{2}{1+2\kappa} n^{(\frac{1}{2}+\kappa)} i^{(-\frac{1}{2}+\kappa)}$. Observe that the lesser the label i of a node, the higher will be its degree. Cooperator nodes with lower labels can suppress defector nodes with higher labels, thereby sustaining cooperation.

THEOREM 8. *Let graph G have vertex set $|V| = \{v_i : 1 \leq i \leq n\}$, and $\forall i, j \in V, Pr[(i,j) \in E] = (ij)^{-\frac{1}{2}+\kappa}$, where $0 < \kappa < \frac{1}{2}$. G sustains cooperation.*

PROOF. Let $0 < \epsilon < 1$ and let ϵ^* take any value $\leq \epsilon$; define $\epsilon' := 1 - \epsilon^*$. Consider α, β, such that $0 < \alpha < \beta < 1$. Let $V_1 = v_i : 1 < i < n^\alpha$; $V_2 = v_i : n^\alpha < i < n^\beta$; and $V_3 = v_i : n^\beta < i < n$. By the Chernoff bound, with high probability there are at least $\frac{1}{2}\epsilon' n^\alpha$ cooperators in V_1. Call this set V_C.

There are at most n^α cooperators in V_1, and hence at least $\frac{n}{2}$ cooperators in $V_2 \uplus V_3$. Call this set V_C'. The expected fitness of any cooperator in V_C – call it f_c – is at least $\sum_{j \in V_C'} \epsilon' (n^\alpha j)^{-\frac{1}{2}+\kappa}$, which is at least $\epsilon' n^{\alpha(-\frac{1}{2}+\kappa)} \sum_{j=n/2}^{n} (j)^{-\frac{1}{2}+\kappa}$ $= \epsilon' n^{\alpha(-\frac{1}{2}+\kappa)}(S(n) - S(n/2))$, where $S(n) = \frac{2}{1+2\kappa} n^{\frac{1}{2}+\kappa}$. Simplifying, we get $f_c \geq 2\gamma n^{\alpha(-\frac{1}{2}+\kappa)+\frac{1}{2}+\kappa}$, where γ is a constant. By the Chernoff bound, we can show that with high probability all cooperators in V_C have fitness higher than $\gamma n^{\alpha(-\frac{1}{2}+\kappa)+\frac{1}{2}+\kappa}$.

Similarly, it can be shown that with high probability each defector in V_3 has fitness at most $\gamma' n^{\beta(-\frac{1}{2}+\kappa)+\frac{1}{2}+\kappa}$. By the above two arguments, with high probability any defector in V_3 who is adjacent to any cooperator in V_C is suppressed (as $\alpha < \beta$).

The probability that a defector in V_3 is not connected to any cooperator in V_C is at most $\prod_{n^\alpha}^{(1-\epsilon')n^\alpha} (1 - (nj)^{-\frac{1}{2}+\kappa})$, which is at most $\gamma e^{-n^{2\kappa}}$, where γ is some constant. The expected number of such defectors is at most $n \cdot \gamma e^{-n^{2\kappa}}$ which goes to zero. Hence with high probability all defectors in V_3 are suppressed, and the number of unsuppressed defectors is at most the size of $|V_1 \uplus V_2|$, which is $o(n)$. $\square$

6. CONCLUSIONS AND FUTURE WORK

We have studied the Evolutionary Prisoner's Dilemma on graphs as a model of cooperation. In particular, we identify the role played by the network topology in sustaining cooperation in a multiagent society. We have shown analytically that for a network to sustain cooperation, it must exhibit properties such as small average diameter, densification, and irregularity. Real-world networks exhibit these properties, and hence could be suitable for sustaining cooperation. Also, we have shown that a family of Scale-Free graphs, a Hierarchy, as well as a family of Bipartite Random graphs sustain cooperation, whereas Erdos-Renyi random graphs do not.

Further exploration along these lines can be carried out to determine which graphs sustain cooperation and which do not, building towards a complete characterization. In our interaction model, we have considered one particular imitation rule (a mutant copies the incumbent strategy if it has an incumbent neighbor with higher fitness) and one particular fitness function (sum of the payoffs received in games played with all neighbors) in the context of the PD game. A similar analysis can be carried out in the context of other strategy update rules and fitness functions, as well as other models of cooperation, such as the Hawk-Dove and Stag-Hunt games.

In general, we feel the kind of analytical treatment carried out in this work yields interesting insights into the factors influencing the sustenance of cooperation, complements the simulation work in the literature, and warrants further studies along similar lines.

7. ACKNOWLEDGMENTS

We thank Dr. Y. Narahari for several key insights and motivations that helped to improve our work.

8. REFERENCES

[1] G. Abramson and M. Kuperman. Social games in a social network. *Physical Review E - Statistical, Nonlinear and Soft Matter Physics*, 63(3 Pt 1):030901, 2001.

[2] R. Axelrod. *The Evolution of Cooperation.* Basic Books, New York, 11 edition, 1984.

[3] R. Axelrod and W. D. Hamilton. The evolution of cooperation. *Science*, 211(4489):1390–1396, 1981.

[4] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[5] F. Chung and L. Lu. The average distance in a random graph with given expected degrees. *Internet Mathematics*, 1(1):91–113, 2003.

[6] O. Durán and R. Mulet. Evolutionary prisoner's dilemma in random graphs. *Physica D: Nonlinear Phenomena*, 208(3-4):10, 2003.

[7] P. Erdos and A. Renyi. On the evolution of random graphs. *Publ. Math. Inst. Hungary. Acad. Sci.*, 5:17–61, 1960.

[8] F. Fu, D. I. Rosenbloom, L. Wang, and M. A. Nowak. Imitation dynamics of vaccination behaviour on social networks. *Proceedings of the Royal Society B: Biological Sciences*, 278(1702):42–49, Jan. 2011.

[9] N. Hanaki, A. Peterhansl, P. S. Dodds, and D. J. Watts. Cooperation in evolving social networks. *Management Science*, 53(7):1036–1050, 2007.

[10] C. Hauert and M. Doebeli. Spatial structure often inhibits the evolution of cooperation in the snowdrift game. *Nature*, 428(6983):643–646, 2004.

[11] L.-M. Hofmann, N. Chakraborty, and K. Sycara. The evolution of cooperation in self-interested agent societies : A critical study. *Proceedings of 10th International Conference on Autonomous Agents and Multiagent Systems*, 2011.

[12] N. Immorlica, B. Lucier, and B. Rogers. Emergence of cooperation in anonymous social networks through social capital. *Proceedings of the 11th ACM Conference on Electronic Commerce*, 2010.

[13] M. S. Kearns and S. Suri. Networks preserving evolutionary equilibria and the power of randomization. In *ACM Conference on Electronic Commerce*, pages 200–207. ACM, 2006.

[14] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, pages 177–187, New York, NY, USA, 2005. ACM.

[15] N. Masuda. Spatial prisoner's dilemma optimally played in small-world networks. *Physics Letters A*, 313(1-2):55–61, 2003.

[16] J. Maynard Smith and G. R. Price. The logic of animal conflict. *Nature*, 246(5427):15–18, 1973.

[17] H. Ohtsuki, C. Hauert, E. Lieberman, and M. A. Nowak. A simple rule for the evolution of cooperation on graphs and social networks. *Nature*, 441(7092):502–505, 2006.

[18] M. J. Osborne. *An Introduction to Game Theory.* Oxford University Press, USA, Aug. 2003.

[19] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation systems. *Communications of the Association for Computing Machinery (CACM)*, 43(12):45–48, 2000.

[20] F. C. Santos and J. M. Pacheco. Scale-free networks provide a unifying framework for the emergence of cooperation. *Physical Review Letters*, 95(9):098104, 2005.

[21] F. C. Santos, J. F. Rodrigues, and J. M. Pacheco. Graph topology plays a determinant role in the evolution of cooperation. *Proceedings of the Royal Society B: Biological Sciences*, 273(1582):51–55, 2006.

[22] G. Szabo and G. Fath. Evolutionary games on graphs. *Physics Reports*, 446(4-6):97–216, 2006.

[23] D. J. Watts and S. H. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393:440–442, 1998.

APPENDIX

A. COMMONLY USED FORMULAE

A.1 Chernoff Bound

Let $(X_1, X_2, ..., X_n)$ be a set of independent random variables, each of which takes value 0 (with probability $1 - p_i$) and value 1 (with probability p_i). Also, let $X = \sum_{i=1}^{n} X_i$, with $\mu = \mathbf{E}[X] = \sum_{i=1}^{n} p_i$, and $0 \leq \lambda \leq 1$. Then,

$$\mathbf{Pr}[X \geq (1+\lambda)\mu] \leq e^{-\frac{\lambda^2}{3}\mu},$$

$$\mathbf{Pr}[X \leq (1-\lambda)\mu] \leq e^{-\frac{\lambda^2}{3}\mu}$$

.

A.2 Large Expectation Lemma

THEOREM 9. *Let $X(n)$ be a non-negative discrete (integer) random variable with density function $p(x)$ and maximum value $M(n)$, with $\mathbf{E}[X(n)] \geq \alpha M(n)$, where $0 \leq \alpha \leq 1$. Let $f(n) = o(M(n))$. Then $\mathbf{Pr}[X \geq f(n)] \geq \alpha/2$.*

PROOF.

$$\mathbf{E}[X(n)] = \sum_{x=0}^{M(n)} xp(x) = \sum_{x=0}^{f(n)-1} xp(x) + \sum_{x=f(n)}^{M(n)} xp(x)$$

$$\sum_{x=f(n)}^{M(n)} M(n)p(x) \geq \sum_{x=f(n)}^{M(n)} xp(x)$$

$$\Rightarrow \mathbf{Pr}[X \geq f(n)] = \sum_{x=f(n)}^{M(n)} p(x) \geq \frac{1}{M(n)} \sum_{x=f(n)}^{M(n)} xp(x)$$

$$\geq \frac{1}{M(n)} \left(\mathbf{E}[X(n)] - \sum_{x=0}^{f(n)-1} xp(x) \right)$$

$$\geq \frac{1}{M(n)} \left(\mathbf{E}[X(n)] - \sum_{x=0}^{f(n)} f(n)p(x) \right)$$

$$\geq \frac{\mathbf{E}[X(n)]}{M(n)} - \frac{f(n)}{M(n)} \sum_{x=0}^{f(n)} p(x)$$

$$\geq \alpha - \frac{f(n)}{M(n)}$$

$$\text{For large enough n, } \frac{f(n)}{M(n)} \leq \frac{\alpha}{2}$$

$$\Rightarrow \mathbf{Pr}[X \geq f(n)] \geq \frac{\alpha}{2}$$

□

Behavioral Game Theoretic Models: A Bayesian Framework For Parameter Analysis

James R. Wright
Department of Computer Science
University of British Columbia
Vancouver, B.C., Canada, V6T 1Z4
jrwright@cs.ubc.ca

Kevin Leyton-Brown
Department of Computer Science
University of British Columbia
Vancouver, B.C., Canada, V6T 1Z4
kevinlb@cs.ubc.ca

ABSTRACT

Studies in experimental economics have consistently demonstrated that Nash equilibrium is a poor description of human players' behavior in unrepeated normal-form games. Behavioral game theory offers alternative models that more accurately describe human behavior in these settings. These models typically depend upon the values of exogenous parameters, which are estimated based on experimental data. We describe methods for deriving and analyzing the posterior distributions over the parameters of such models, and apply these techniques to study two popular models (Poisson-CH and QLk), the latter of which we previously showed to be the best-performing existing model in a comparison of four widely-studied behavioral models [22]. Drawing on a large set of publicly available experimental data, we derive concrete recommendations for the parameters that should be used with Poisson-CH, contradicting previous recommendations in the literature. We also uncover anomalies in QLk that lead us to develop a new, simpler, and better-performing family of models.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Economics, Experimentation, Performance

Keywords

Behavioral game theory, Bounded rationality, Game theory, Cognitive models

1. INTRODUCTION

It is well known that in many settings, the standard game-theoretic assumption that agents will adopt *Nash equilibrium* strategies—where each agent simultaneously responds optimally to all the others—is a poor predictor of actual human behavior [e.g., see 9]. The field of *behavioral game theory* aims to develop models that more accurately describe human behavior, as evaluated using experimental data [2]. These models typically depend upon parameters such as agents' sensitivity to utility differences and the distribution of cognitive ability in the population. In order for behavioral models to be effective tools for prediction, it is necessary to identify "good" estimates for these parameters. Such parameter values are also interesting for their own sake, because they can offer economic insights into human behavior.

In past work, we conducted an exhaustive meta-study of behavioral game theory models, focusing on the problem of unrepeated, simultaneous-move games [22]. We evaluated four prominent models from the behavioral game theory literature: Quantal Response Equilibrium [QRE; 12], Level-k [Lk; 13, 6], Poisson–Cognitive Hierarchy [Poisson-CH; 3], and Quantal Level-k [QLk; 19]. We evaluated these models according to their out-of-sample prediction performance: we identified parameters for each model that maximized the likelihood of a training data set, and then evaluated each model in terms of the likelihood it assigned to a separate test set. We took data from six publicly available datasets describing lab experiments in which subjects played unrepeated, simultaneous-move games against other human opponents. In the end we observed a result that was statistically significant and quite robust: that the QLk model substantially outperformed all others on the set of all data, and also had the best or nearly the best performance on each individual dataset.

While it effectively identified the best-performing existing model, our previous work offered very little insight about the models' parameter values. For example, because we relied upon a maximum-likelihood approach, we obtained no information about the extent to which parameter values could be changed without a major drop in predictive accuracy, or even about the extent to which individual parameters influence a model's performance. This paper shows how to answer such questions. Specifically, it describes a Bayesian approach for gaining understanding about a behavioral model's entire parameter space. We combine experimental data with explicitly quantified prior beliefs to derive a posterior distribution that assigns probability to parameter settings in proportion to their consistency with the data and the prior [8]. We apply this approach to analyze the posterior distributions for two models: QLk and Poisson–Cognitive Hierarchy. Although Poisson-CH did not demonstrate competitive performance in [22], we analyze it here because it is very simple, and because the literature contains a very specific recommendation for how its parameter should be set: Camerer et al. [3] recommend setting the model's single parameter, which represents agents' mean number of steps of strategic

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

reasoning, to 1.5. Our own analysis sharply contradicts this recommendation, placing the 99% confidence interval almost a factor of three lower, on the range $[0.51, 0.59]$. We devote most of our attention to QLk, however, since we previously showed it to achieve strong performance. Our new analysis points out several anomalies in the parameter distributions for QLk, suggesting that a simpler model could be preferable. By exhaustively evaluating a family of variations on QLk, we identify a simpler, more predictive model based in part on the cognitive hierarchy concept.

We pause to contrast our work with a particularly related paper from the economics community. Rogers et al. [17] proposed a unifying framework that generalizes both Poisson-CH and QRE, and compared the fit of several variations within this framework. This is similar to our search of a family of QLk variants, but there are several differences. First, we compare out-of-sample prediction performance, not in-sample fit. Second, Rogers et al. restricted the distributions of types to be grid, uniform, or Poisson distributions, whereas we consider unconstrained discrete distributions. Third, they required different types to have different precisions, while we do not. Finally, we consider level-k beliefs as well as cognitive hierarchy beliefs, whereas they compared only cognitive hierarchy belief models (although their framework in principle allows for both).

There has also been interest by various computer scientists in behavioral solution concepts as an alternative to standard economic models. For example, Wunder et al. [23] extended the cognitive hierarchy model in their analysis of the Lemonade Stand Game, and Gao and Pfeffer [7] used quantal response equilibrium to model bounded rationality in a game where agents respond to payoffs unknown to the modeler.

In the next section, we define the models that we analyze. We then describe the framework that we used for estimating and analyzing parameters and their distributions, and the dataset and specific techniques that we used to derive our estimates. In Section 4 we present the results of our analyses for the Poisson-CH and QLk models. In Section 5, we describe the QLk-like variants that we evaluated, and present the results of our search.

2. BEHAVIORAL MODELS

We begin by formally defining the behavioral models that we analyze. We then relate these models to other widely studied models that we do not consider further, due to their poor predictive performance in our previous study [22].

2.1 Quantal Level-k

Stahl and Wilson [19] proposed a rich model of strategic reasoning that we refer to as the *quantal level-k model* (QLk). The QLk model of human behavior incorporates two primary components: *iterated strategic reasoning*, and *cost-proportional errors*. Iterated strategic reasoning refers to a limit on the number of levels of higher-order belief that agents can maintain.[1] Agents make cost-proportional errors if their rate of making errors increases as errors become less costly. This can be modeled by assuming that agents best respond *quantally*, rather than via strict maximization.

Definition (Quantal best response). Let $u_i(a_i, s_{-i})$ be agent i's expected utility when playing action a_i against strategy profile s_{-i}. Then a *quantal best response* $QBR_i(s_{-i}; \lambda)$ by agent i to s_{-i} is a mixed strategy s_i such that

$$s_i(a_i) = \frac{\exp[\lambda \cdot u_i(a_i, s_{-i})]}{\sum_{a'_i} \exp[\lambda \cdot u_i(a'_i, s_{-i})]}, \tag{1}$$

where λ (the *precision*) indicates agents' sensitivity to utility differences. $\square$

Note that unlike regular best response, which is a set-valued function, quantal best response always returns a single mixed strategy. When $\lambda = 0$, quantal response mixes uniformly over all of the agents' actions; as $\lambda \to \infty$, quantal best response approaches strict best response.

In the QLk model, agents have one of three levels: level-0, level-1, or level-2. Level-0 agents are nonstrategic, and choose their actions uniformly at random. Level-1 agents quantally best respond to level-0 agents; that is, they believe that the rest of the population consists entirely of level-0 agents. Similarly, level-2 agents quantally respond to level-1 agents. Level-1 and level-2 agents can use different precisions (λ's), and furthermore level-2 agents' beliefs about level-1 agents' precision can be arbitrarily different from level-1 agents' actual precision.

Definition (QLk model). Let A_i denote player i's action set. Then the probability distribution $\pi^{QLk}_{i,k} \in \Pi(A_i)$ over actions that QLk predicts for a level-k agent playing as agent i is defined as follows.

$$\begin{aligned} \pi^{QLk}_{i,0}(a_i) &= |A_i|^{-1}, \\ \pi^{QLk}_{i,1} &= QBR_i(\pi^{QLk}_{-i,0}; \lambda_1), \\ \pi^{QLk}_{i,1(2)} &= QBR_i(\pi^{QLk}_{-i,0}; \lambda_{1(2)}), \\ \pi^{QLk}_{i,2} &= QBR_i(\pi^{QLk}_{i,1(2)}; \lambda_2), \end{aligned}$$

where $\pi^{QLk}_{i,1(2)}$ is a mixed-strategy profile representing level-2 agents' (possibly incorrect) beliefs about how level-1 agents play. The overall predicted distribution π^{QLk} of actions is the weighted sum of the distributions for each level: $\pi^{QLk}_i = \sum_{k=0}^{2} \alpha_k \pi^{QLk}_{i,k}$, where $\alpha_0 = 1 - \alpha_1 - \alpha_2$. The QLk model thus has five parameters: $\{\alpha_1, \alpha_2\}$, the proportions of level-1 and level-2 agents ($\alpha_1 + \alpha_2 \leq 1$; any remaining agents are level-0); $\{\lambda_1, \lambda_2\}$, the precisions of level-1 and level-2 agents' responses; and $\lambda_{1(2)}$, level-2 agents' beliefs about the precision of level-1 agents. $\square$

2.2 Poisson Cognitive Hierarchy

Like QLk, the cognitive hierarchy model [3] aims to model agents with heterogeneous bounds on iterated reasoning. It differs from the QLk model in two key ways. First, agents use standard best response, rather than quantal response. Second, agents best respond to the full distribution of lower-level types, rather than only to the strategy one level below. More formally, every agent has an associated level $m \in \{0, 1, 2, \ldots\}$. Let f be a probability mass function describing the distribution of the levels in the population. Level-0 agents play uniformly at random. Level-m agents ($m \geq 1$) best respond to the strategies that would be played in a population described by the truncated probability mass function $f(j \mid j < m)$.

Camerer et al. [3] advocate a single-parameter restriction of the cognitive hierarchy model called *Poisson-CH*, in which f is a Poisson distribution.

[1]This limit is believed to be quite low; e.g., Arad and Rubinstein [1] found no support for 4th order or higher beliefs.

Definition (Poisson-CH model). Let $\pi_{i,m}^{PCH} \in \Pi(A_i)$ be the distribution over actions predicted for an agent i with level m by the Poisson-CH model. Let $f(m) = \text{Poisson}(m; \tau)$. Let $BR_i(s_{-i})$ denote the set of i's best responses to the strategy profile s_{-i}. Let $\pi_{i,0:m}^{PCH} = \sum_{\ell=0}^{m} f(\ell)\pi_{i,\ell}^{PCH} / \sum_{\ell=0}^{m} f(\ell)$ be the "truncated" distribution over actions predicted for an agent conditional on that agent's having level $0 \leq \ell \leq m$. Then π^{PCH} is defined as follows:

$$\pi_{i,0}^{PCH}(a_i) = |A_i|^{-1},$$

$$\pi_{i,m}^{PCH}(a_i) = \begin{cases} |BR_i(\pi_{i,0:m-1}^{PCH})|^{-1} & \text{if } a_i \in BR_i(\pi_{i,0:m-1}^{PCH}), \\ 0 & \text{otherwise.} \end{cases}$$

As with QLk, the overall predicted distribution of actions π^{PCH} is a weighted sum of the distributions for each level: $\pi_i^{PCH} = \sum_{\ell=0}^{\infty} f(\ell)\pi_{i,\ell}^{PCH}$. The mean of the Poisson distribution, τ, is this model's single parameter. □

2.3 Other Behavioral Models

We previously evaluated two other behavioral models in addition to those listed above [22]. The first, QRE [12], is a generalization of Nash equilibrium where agents respond quantally instead of best responding. The second, Lk [13, 6], is similar to QLk, except that agents best respond to the next level down rather than quantally responding. We omit these models from our analysis, since they consistently achieved worse predictive performance than QLk in our previous comparison.

3. BAYESIAN PARAMETER ANALYSIS

In this section, we describe this paper's first contribution, our formal framework for parameter estimation, along with the experimental data and estimation techniques we used.

3.1 Prediction Framework

We begin by noting that any behavioral game theoretic model is a mapping from a game description G and a vector of parameters θ to a distribution over action profiles in G. In other words, a given behavioral model provides us with a way of computing $\Pr(a \mid G, \theta)$. Our goal is to use this behavioral model in conjunction with our experimental data, $\mathcal{D}$, to predict future behavior in a given game G. The dataset may or may not contain examples of earlier play in G. The model parameters need to be determined based upon the data. There are two approaches that we can take.

The frequentist approach, taken by Rogers et al. [17] and Wright and Leyton-Brown [22], is to compute a point estimate $\hat{\theta}$ of the model's parameters based on our dataset, and then use $\hat{\theta}$ to compute the prediction for new games. That is, we use the model

$$\Pr(a \mid G, \mathcal{D}) = \Pr(a \mid G, \hat{\theta}). \qquad (2)$$

The Bayesian approach is to calculate a posterior distribution over parameter values $\Pr(\theta \mid \mathcal{D})$, and then use this distribution to integrate out the parameter. This yields the alternate model

$$\Pr(a \mid G, \mathcal{D}) = \int \Pr(a \mid G, \theta) \Pr(\theta \mid \mathcal{D}) d\theta. \qquad (3)$$

Observe that Equation (2) can be seen as a special case of (3), where the posterior is a point mass at $\hat{\theta}$.

We use the posterior distribution to analyze model parameters, which is an application of Bayesian estimation. However, on our dataset, we have observed that the frequentist approach achieves better predictive performance than the Bayesian approach. Thus, we use point estimation to compare performance between model variants. We describe both approaches.

3.1.1 *Point Estimation*

We use the maximum likelihood estimate of the parameters as our point estimate:

$$\hat{\theta} = \arg\max_{\theta} \Pr(\mathcal{D} \mid \theta).$$

Our dataset consists of a set $\mathcal{D}$ of observations $d_i = (G_i, a_i)$, indicating that an action profile a_i was played in game G_i. The likelihood of a single datapoint is

$$\Pr(d_i \mid \theta) = \Pr(G_i, a_i \mid \theta).$$

By the chain rule (of probabilities), this is equivalent to

$$\Pr(d_i \mid \theta) = \Pr(a_i \mid G_i, \theta) \Pr(G_i \mid \theta). \qquad (4)$$

We assume that θ and G are independent; i.e., we assume that $\Pr(G_i \mid \theta) = \Pr(G_i)$. Intuitively, this means that we assume there is a single "true" parameter value θ^* for all games, rather than a separate "true" θ^{G_i} for each game G_i. This allows us to rewrite (4) as

$$\Pr(d_i \mid \theta) = \Pr(a_i \mid G_i, \theta) \Pr(G_i). \qquad (5)$$

We assume that the datapoints are independent, so the likelihood of the dataset is just the product of the likelihoods of the datapoints:

$$\Pr(\mathcal{D} \mid \theta) = \prod_{d_i \in \mathcal{D}} \Pr(d_i \mid \theta). \qquad (6)$$

Substituting (5) into (6) gives us

$$\Pr(\mathcal{D} \mid \theta) = \prod_{d_i \in \mathcal{D}} \Pr(a_i \mid G_i, \theta) \Pr(G_i). \qquad (7)$$

The probabilities $\Pr(G_i)$ are constant with respect to θ, and can therefore be disregarded when maximizing likelihood:

$$\arg\max_{\theta} \Pr(\mathcal{D} \mid \theta) = \arg\max_{\theta} \prod_{d_i \in \mathcal{D}} \Pr(a_i \mid G_i, \theta).$$

3.1.2 *Bayesian Estimation*

We derive an expression for the posterior distribution $\Pr(\theta \mid \mathcal{D})$ by applying Bayes' rule, where $p_0(\theta)$ is the prior:

$$\Pr(\theta \mid \mathcal{D}) = \frac{p_0(\theta) \Pr(\mathcal{D} \mid \theta)}{\Pr(\mathcal{D})}. \qquad (8)$$

Substituting in (7), the posterior distribution is

$$\Pr(\theta \mid \mathcal{D}) = \frac{p_0(\theta) \prod_{d_i \in \mathcal{D}} \Pr(a_i \mid G_i, \theta) \Pr(G_i)}{\Pr(\mathcal{D})},$$

and since both $\Pr(G_i)$ and $\Pr(\mathcal{D})$ are constant with respect to θ,

$$\Pr(\theta \mid \mathcal{D}) \propto p_0(\theta) \prod_{d_i \in \mathcal{D}} \Pr(a_i \mid G_i, \theta). \qquad (9)$$

Codename	Source	Games	n	Units
SW94	Stahl and Wilson [19]	10	400	\$0.025
SW95	Stahl and Wilson [20]	12	576	\$0.02
CGCB98	Costa-Gomes et al. [5]	18	1566	\$0.022
GH01	Goeree and Holt [9]	10	500	\$0.01
CVH03	Cooper and Huyck [4]	8	2992	\$0.10
RPC09	Rogers et al. [17]	17	1210	\$0.01
HSW01	Haruvy et al. [11]	15	869	\$0.02
HS07	Haruvy and Stahl [10]	20	2940	\$0.02
SH08	Stahl and Haruvy [18]	18	1288	\$0.02
COMBO9	Above datasets	128	3600	\$0.01

Table 1: Names and contents of each dataset. The column headed n indicates the number of observations in the dataset. Units are in expected value.

3.2 Data

We analyzed data from nine experimental studies, summarized in Table 1; this expands beyond the six studies we considered in our previous work [22]. We also constructed a new dataset (COMBO9) containing observations from all nine source datasets. To ensure that each was equally represented, despite their differences in size, we included exactly 400 observations from each dataset (sampled uniformly without replacement).

The precision parameter for quantal response is not *scale invariant.* That is, the correct value of λ can differ depending upon the units in which payoffs are expressed. To ensure consistent estimation of precision parameters, we renormalized all games so that their payoffs were in expected cents.

3.3 Estimation Methods

For all of the models we considered, we used a flat prior for the parameters. Although this prior is improper, it results in a correctly-normalized posterior distribution; the posterior distribution in this case reduces to the likelihood [8]. We computed the posterior distribution for the single-parameter Poisson-CH model by *grid sampling.* That is, we computed the likelihood of the COMBO9 dataset for each value of $\tau \in \{0.01k \mid k \in \mathbb{N}, 0 \leq 0.01k \leq 10\}$, and then normalized by the sum of the likelihoods.

The QLk model has five parameters, and is therefore too high dimensional to grid sample efficiently; approximately 5×10^{12} samples would have been required for a grid of the same granularity as we used for Poisson-CH! Instead, for QLk—and the other high dimensional models introduced in Section 5—we used a sequential Monte Carlo technique called *annealed importance sampling*, or AIS [14]. AIS allows for efficient sampling from high dimensional distributions, similarly to Markov Chain Monte Carlo (MCMC) techniques. However, each sample point generated using AIS is independent, so AIS does not exhibit the random-walk behavior that can plague MCMC samplers. Briefly, the annealed importance sampling procedure is as follows. A sample $\vec{\theta}_0$ is drawn from an easy-to-sample-from distribution P_0. For each P_j in a sequence of intermediate distributions $P_1, \ldots, P_{r-1}$ that become progressively closer to the posterior distribution, a sample $\vec{\theta}_j$ is generated by drawing a sample $\vec{\theta}'$ from a *proposal distribution* $Q(\cdot \mid \vec{\theta}_{j-1})$, and accepted with probability

$$\frac{P_j(\vec{\theta}')Q(\vec{\theta}_{j-1} \mid \vec{\theta}')}{P_j(\vec{\theta}_{j-1})Q(\vec{\theta}' \mid \vec{\theta}_{j-1})}. \tag{10}$$

If the proposal is accepted, $\vec{\theta}_j = \vec{\theta}'$; otherwise, $\vec{\theta}_j = \vec{\theta}_{j-1}$. We repeat this procedure multiple times, and report the distribution of the resulting $\vec{\theta}_r$ values, with each $\vec{\theta}_r$'s contribution weighted according to

$$\frac{P_1(\vec{\theta}_0)P_2(\vec{\theta}_1)}{P_0(\vec{\theta}_0)P_1(\vec{\theta}_1)} \cdots \frac{P_{r-1}(\vec{\theta}_{r-2})P_r(\vec{\theta}_{r-1})}{P_{r-2}(\vec{\theta}_{r-2})P_{r-1}(\vec{\theta}_{r-1})}. \tag{11}$$

For the initial sampling distribution P_0, we used a product distribution over the population proportions parameters and the precision parameters. For the population proportion parameter components we used a Dirichlet distribution $\mathrm{Dir}(1,1,1)$; this is equivalent to uniformly sampling over the simplex of all possible combinations of population proportions. For the precision parameter components we used the renormalized non-negative half of a univariate Gaussian distribution $\mathcal{N}(0, 2^2)$ for each precision parameter; this gives a distribution that is decreasing in precision (on the assumption that higher precisions are less likely than lower ones), and with a standard deviation of 2, which was large enough to give a non-negligible probability to most previous precision estimates.

The proposal distribution was a product distribution "centered" at the current value, with proportion parameters $\vec{\alpha}'$ sampled from $\mathrm{Dir}(20\vec{\alpha}_{j-1})$, and each precision parameter λ' sampled from $\mathcal{N}(\lambda_{j-1}, 0.2^2)$ (truncated at 0 and renormalized). The "hyperparameters" for the Dirichlet distribution (20) and the precision distributions (0.2^2) were chosen by trial and error on a small subset of the data to make the acceptance rate near to the standard heuristic value of 0.5 [16]. We used 200 intermediate distributions of the form

$$P_j(\vec{\theta}) = \Pr(\vec{\theta} \mid \mathcal{D})^{\gamma_j},$$

with the first 40 γ_j's spaced uniformly from 0 to 0.01, and the remaining 160 γ_j's spaced geometrically from 0.01 to 1, as in the original AIS description [14]. We performed 5 Metropolis updates in each distribution before moving to the next distribution in the chain.

For the model performance comparisons in Section 5, we computed the parameter point-estimates using the Nelder-Mead simplex algorithm [15]. To reduce the danger of getting caught in a local minimum, we repeated the optimization from 500 random starting points, and chose the value which gave the highest log likelihood.

4. RESULTS

In this section, we describe the results of our analysis of posterior distributions for the two models defined in Section 2. We start by comparing the posterior distribution for the Poisson-CH model's parameter to a previously published recommendation. We then turn our attention to the QLk model, whose posterior distribution exposes several possible issues with the model.

4.1 Poisson-CH

Camerer et al. [3] recommended setting the τ parameter of the Poisson-CH model to 1.5. Figure 1 gives the cumulative posterior distribution over τ for each of our datasets. Overall, our analysis strongly contradicts Camerer et al.'s recommendation. On COMBO9, the posterior probability of $0.51 \leq \tau \leq 0.59$ is more than 99%. Every other source dataset had a wider high posterior density region than COMBO9 (indicated by the higher slope of COMBO9's cumulative density

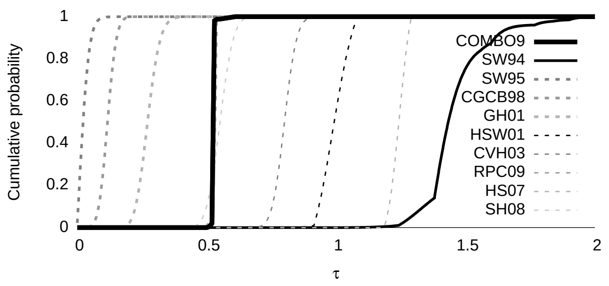


Figure 1: Cumulative posterior distributions for the τ parameter of the Poisson-CH model. Bold trace is for the combined dataset; solid trace is for the outlier Stahl and Wilson [19] source dataset; dotted traces are all other source datasets.

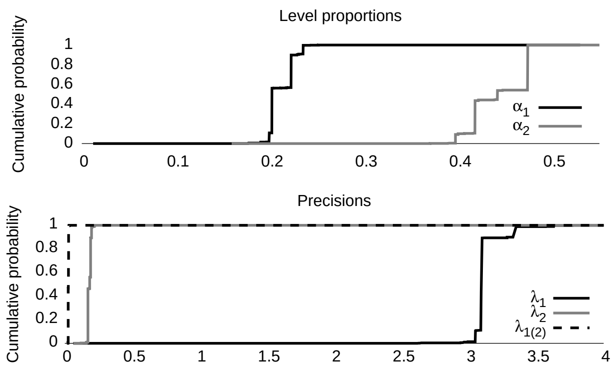


Figure 2: Marginal cumulative posterior distribution for the level proportion parameters (α_1, α_2; top panel) and precision parameters ($\lambda_1, \lambda_2, \lambda_{1(2)}$; bottom panel) of the QLk model on the combined dataset.

function); this is expected, as smaller datasets lead to less confident predictions. Nevertheless, all but two of the source datasets had median values less than 1.0. Only the Stahl and Wilson [19] dataset (SW94) appears to support Camerer et al.'s recommendation (median 1.43). However, SW94 appears to be an outlier; its high posterior density region is wider than the other distributions, and the distribution is very multimodal, likely due to SW94's small size.

4.2 QLk

Figure 2 gives the marginal cumulative posterior distributions for each of the parameters of the QLk model. (That is, we computed the five-dimensional posterior distribution, and then extracted from it the five marginal distributions shown here.) We found these distributions surprising for several reasons. First, the models predict many more level-2 agents than level-1 agents. In contrast, it is typically assumed that higher level agents are scarcer, as they perform more complex strategic reasoning. Even more surprisingly, the model predicts that level-1 agents should have much higher precisions than level-2 agents. This is odd if the level-2 agents are to be understood as "more rational"; indeed, precision is sometimes interpreted as a measure of rationality [e.g., see 21, 7]. Third, the distribution of $\lambda_{1(2)}$, the precision that level-2 agents ascribe to level-1 agents, is very concentrated around very small values ($[0.023, 0.034]$). This differs by two orders of magnitude from the "true" value of λ_1, which is quite concentrated around its median value of 3.1. Finally, the median value of λ_1 (3.1) is more than 17 times larger than that of λ_2 (0.18). It seems unlikely that level-1 agents would be an order of magnitude more sensitive to utility differences than level-2 agents.

One interpretation is that the QLk model is essentially accurate, and these parameter values simply reflect a surprising reality. For example, the low precision of level-2 agents and the even lower precision that they (incorrectly) ascribe to the level-1 agents may indicate that two-level strategic reasoning causes a high cognitive load, which makes agents more likely to make mistakes, both in their own actions and in their predictions. The main appeal of this explanation is that it allows us to accept the QLk model's strong performance at face value.

An alternate interpretation is that QLk fails to capture some crucial aspect of experimental subjects' strategic reasoning. For example, if the higher-level agents reasoned about all lower levels rather than only one level below themselves, then the low value of $\lambda_{1(2)}$ could predict well because it "simulates" a model where level-2 agents respond to a mixture of level-0 and level-1 agents. We investigate this second possibility in the next section.

5. MODEL VARIATIONS

In this section, we investigate the properties of the QLk model by evaluating the predictive power of a family of systematic variations of the model. In the end, we identify a simpler model that dominates QLk on our data, and which also yields much more reasonable marginal distributions over parameter values.

Specifically, we constructed a family of models by extending or restricting the QLk model along four different axes. QLk assumes a maximum level of 2; we varied this by considering maximum levels of 1 and 3 as well. QLk has *inhomogeneous precisions* in that it allows each level to have a different precision; we varied this by also considering *homogeneous precision* models. QLk allows *general precision beliefs* that are not restricted to be accurate; we also constructed models that make the simplifying assumption of *accurate precision beliefs* about lower levels' precisions. Finally, in addition to *Lk* beliefs, where all other agents are assumed by a level-k agent to be level-$(k-1)$, we also constructed models with *CH* beliefs, where agents believe that the population consists of the true, truncated distribution over the lower levels. We evaluated each combination of axis values; the 17 resulting models[2] are listed in the first part of Table 2. In addition to the 17 exhaustive axis combinations for models with maximum levels in $\{1, 2, 3\}$, we also evaluated 12 additional axis combinations that have higher maximum levels and 8 parameters or fewer: `ai-QCH4` and `ai-QLk4`; `ah-QCH` and `ah-QLk` variations with maximum levels in $\{4, 5, 6, 7\}$; and `ah-QCH` and `ah-QLk` variations that assume a Poisson distribution over the levels rather than using an explicit tabular distribution. These additional models are listed in the bottom part of Table 2.

5.1 Simplicity Versus Predictive Performance

[2]When the maximum level is 1, all combinations of the other axes yield identical predictions. Therefore there are only 17 models instead of $3 \cdot 2^3 = 24$.

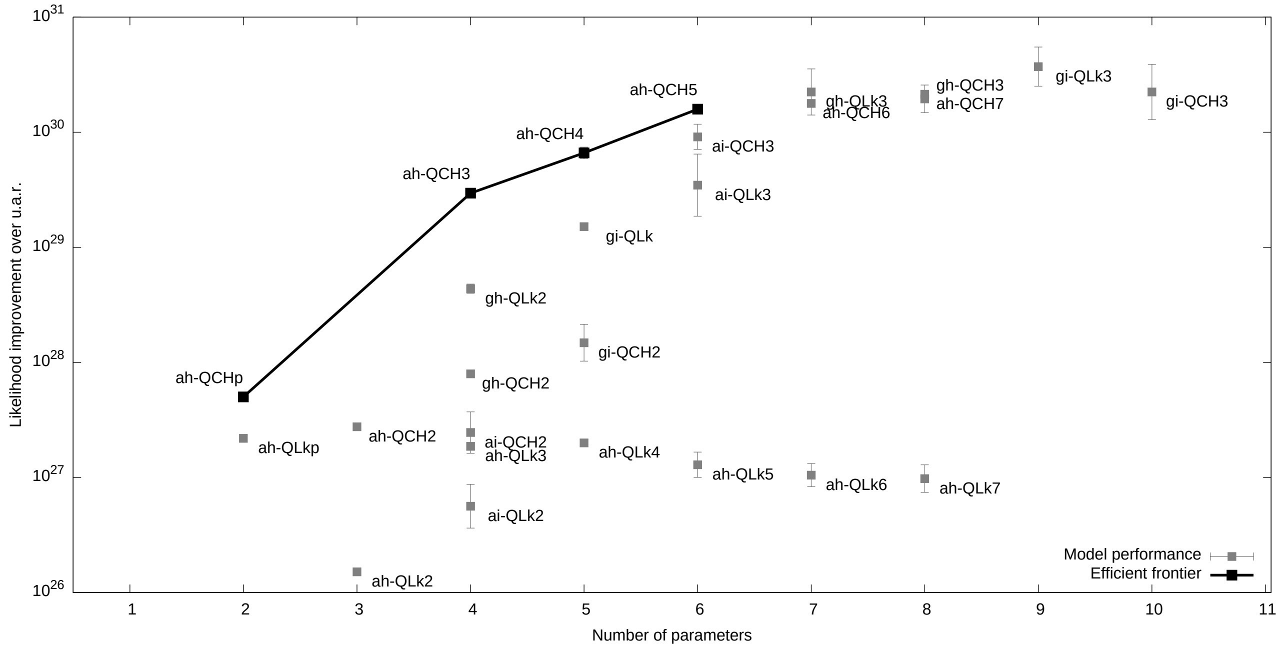


Figure 3: Model simplicity (number of parameters) versus prediction performance. `QLk1`, which has far lower performance than the other models, is omitted for scaling reasons.

We evaluated the predictive performance of each model on the COMBO9 dataset using 10-fold cross-validation repeated 10 times, as in [22]. The results are given in the last column of Table 2, and plotted in Figure 3. Performance is measured by how much more likely the test data is according to the given model (with parameters chosen based on a training dataset, separate from the test set) than it is according to assuming that actions are chosen uniformly at random (u.a.r.). E.g., the test data is $10^{18.37}$ times more likely according to the `QLk1` model's prediction than according to a uniform random prediction.

All else being equal, a model with higher performance is more desirable, as is a model with fewer parameters. We can plot an "efficient frontier" of models with the (statistically significant) highest performance for a given number of parameters or fewer; see Figure 3. The original QLk (`gi-QLk2`) is *not* efficient in this sense; it is dominated by `ah-QCH3`, which has significantly better predictive performance even though it has fewer parameters (due to restricting agents to homogeneous precisions and accurate beliefs). Our analysis thus argues that the flexibility added by inhomogeneous precisions and general precision beliefs is less important than the number of levels and the choice of population belief.

Conversely, the poor performance of the Poisson variants relative to `ah-QCH3` suggests that flexibility in describing the level distribution is more important than the total number of levels modeled. Figure 4 shows the marginal posterior level weight distributions for all four models on the efficient frontier. There is broad agreement among all models on the proportion of level-0 agents. However, in order to get the "right" number of level-0 agents, `ah-QCHp` (which models the level distribution as a Poisson) must place a great deal of weight on level-1 agents as well; in contrast, the tabular-distribution models all select bimodal distributions that assign relatively little weight to level-1 agents, and more to higher-level agents (level-2 and higher).

In fact, there is a pattern in the models along the efficient frontier: this set consists *exclusively* of models with accurate precision beliefs, homogeneous precisions, and cognitive hierarchy beliefs.[3] This suggests that the most parsimonious way to model human behavior in normal-form games is to use a model of this form, with the tradeoff between simplicity (i.e., number of parameters) and predictive power determined solely by the number of levels modeled.

To test the extent of the simplicity/power tradeoff in this family of models, we also evaluated `ah-QCH4`, `ah-QCH5`, `ah-QCH6`, and `ah-QCH7`, and for comparison purposes also evaluated every other model in our design space having 8 or fewer parameters. For the COMBO9 dataset, adding additional levels yielded small increases in predictive power until level 5, after which it yielded no further, statistically significant improvements. Thus, Figure 3 includes `ah-QCH4` and `ah-QCH5` as part of the efficient frontier.

Overall, we recommend that practitioners seeking a model for predicting human behavior in simultaneous-move games use a member of the `ah-QCH` family. How to model the distribution of levels is less clear. Of the models we considered, we recommend `ah-QCH3`. While it is possible to achieve gains in performance on our (large) dataset by modeling additional levels, these gains are small. Specifically, the gain in

[3]One might be interested in a weaker definition of the efficient frontier, saying that a model is efficient if it achieves significantly better performance than all *efficient* models with fewer parameters, rather than *all* models with fewer parameters. In this case the efficient frontier consists of all models previously identified as efficient plus `ah-QCH7` and `gi-QLk3`. Our original definition rejected `gi-QLk3` because it did not predict significantly better than `gh-QLk3`, which in turn did not predict significantly better than `ah-QCH5`.

Name	Max Level	Pop'n Beliefs	Precisions	Prec. Beliefs	#	Log-likelihood vs. u.a.r.
QLk1	1	n/a	n/a	n/a	2	18.37 ± 0.12
gi-QLk2	2	Lk	inhomo.	general	5	29.18 ± 0.03
ai-QLk2	2	Lk	inhomo.	accurate	4	26.75 ± 0.19
gh-QLk2	2	Lk	homo.	general	4	28.64 ± 0.04
ah-QLk2	2	Lk	homo.	accurate	3	26.18 ± 0.03
gi-QCH2	2	CH	inhomo.	general	5	28.17 ± 0.16
ai-QCH2	2	CH	inhomo.	accurate	4	27.39 ± 0.18
gh-QCH2	2	CH	homo.	general	4	27.90 ± 0.03
ah-QCH2	2	CH	homo.	accurate	3	27.44 ± 0.02
gi-QLk3	3	Lk	inhomo.	general	9	30.57 ± 0.17
ai-QLk3	3	Lk	inhomo.	accurate	6	29.54 ± 0.27
gh-QLk3	3	Lk	homo.	general	7	30.35 ± 0.20
ah-QLk3	3	Lk	homo.	accurate	4	27.27 ± 0.03
gi-QCH3	3	CH	inhomo.	general	10	30.35 ± 0.24
ai-QCH3	3	CH	inhomo.	accurate	6	29.96 ± 0.11
gh-QCH3	3	CH	homo.	general	8	30.29 ± 0.12
ah-QCH3	3	CH	homo.	accurate	4	29.47 ± 0.02
ai-QLk4	4	Lk	inhomo.	accurate	8	30.05 ± 0.26
ah-QLk4	4	Lk	homo.	accurate	5	27.30 ± 0.03
ah-QLk5	5	Lk	homo.	accurate	6	27.11 ± 0.11
ah-QLk6	6	Lk	homo.	accurate	7	27.02 ± 0.10
ah-QLk7	7	Lk	homo.	accurate	8	26.99 ± 0.12
ah-QLkp	*	Lk	homo.	accurate	2	27.34 ± 0.02
ai-QCH4	4	CH	inhomo.	accurate	8	29.86 ± 0.20
ah-QCH4	4	CH	homo.	accurate	5	29.82 ± 0.05
ah-QCH5	5	CH	homo.	accurate	6	30.20 ± 0.04
ah-QCH6	6	CH	homo.	accurate	7	30.25 ± 0.03
ah-QCH7	7	CH	homo.	accurate	8	30.33 ± 0.02
ah-QCHp	*	CH	homo.	accurate	2	27.70 ± 0.02

Table 2: Model variations, evaluated on the Combo9 dataset. The column headed # indicates the number of parameters in the model; the models with max level of $*$ used a Poisson distribution.

prediction from adding two parameters to ah-QCHp to yield ah-QCH3 was more than twice as great as was the gain in prediction from adding two parameters to ah-QCH3 to yield ah-QCH5. Since more complex models are more prone to over-fitting—particularly with smaller amounts of data—we believe that ah-QCH3 offers the best tradeoff between robustness and experimental performance. A practitioner working with a large dataset and interested in maximal prediction quality might also investigate ah-QCH5. Another possibility, which we have begun to explore in our current work, is to use a parametric distribution of levels whose shape is a better match to the experimental data than the Poisson distribution.

5.2 Parameter Analysis for ah-QCH3

We are now in a position to answer some of the questions from Section 4.2 by examining marginal posterior distributions from a member of our new model family, plotted in Figure 5.

We first note that, in contrast to QLk's multimodal, jagged parameter CDFs, the parameter CDFs for ah-QCH3 are smooth and (nearly) unimodal. This suggests that ah-QCH3 is a much more robust model; its prediction quality is less likely to change drastically as a result of small changes in parameter values.

Second, the posterior distribution for the precision parameter λ is concentrated around 0.20, which is very close to the QLk model's estimate for λ_2. This suggests that QLk's much lower estimate for $\lambda_{1(2)}$ may have been the closest that the

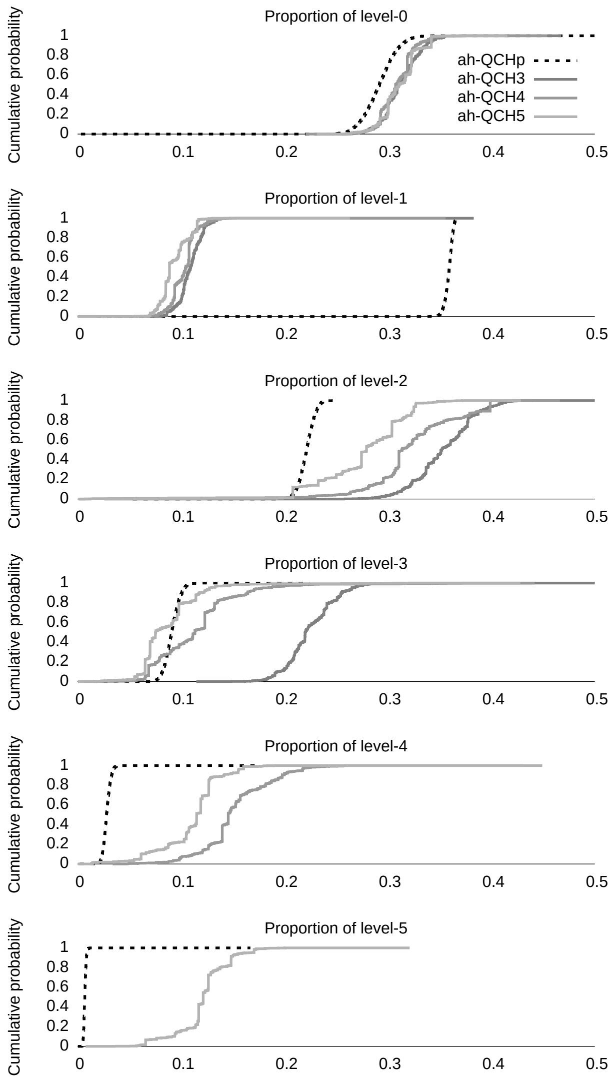


Figure 4: Marginal cumulative posterior distributions of levels of reasoning for efficient frontier models.

model could get to having the level-2 agents best respond to a mixture of level-0 and level-1 agents (as in cognitive hierarchy). It is unclear whether the order-of-magnitude differences and counterintuitive ordering of λ_1 and λ_2 are similar effects where QLk's parameters are set in a way that "simulates" the assumptions of a more accurate model. Interestingly, like QLk, the ah-QCH3 model predicts more level-2 agents than level-1. In fact, the ah-QCH3 model predicts even fewer level-1 agents than QLk. This provides some support for QLk's seemingly counterintuitive prediction that level-1 agents are less common than more sophisticated types.

6. CONCLUSIONS

We showed how Bayesian parameter analysis can be used to gain understanding about the sensitivity of behavioral game theoretic models to their parameters. We derived concrete recommendations for the use of an existing model, Poisson-CH, which differed substantially from advice in the

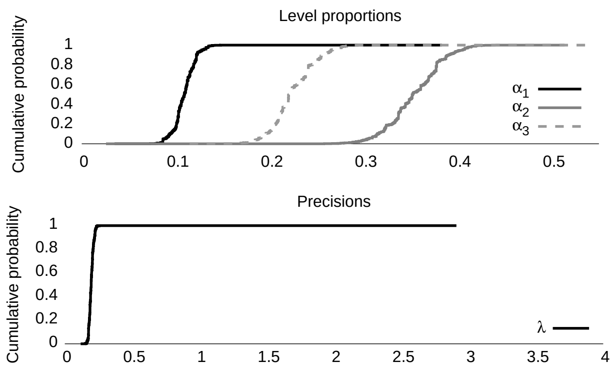


Figure 5: Marginal cumulative posterior distributions for the level proportion parameters ($\alpha_1, \alpha_2, \alpha_3$; top panel) and precision parameter (λ; bottom panel) of the `ah-QCH3` model on the combined dataset.

literature. We also uncovered anomalies in the best-performing existing model (QLk) that led us to a new, simpler, better-performing model (`ah-QCH3`). More broadly, the family of accurate, homogeneous-precision QCH models allows the modeler to trade off complexity against performance along an efficient frontier of models simply by adjusting the model of the distribution of levels.

In our ongoing work, we are now investigating parametric distributions of levels that match the observed distribution of levels. This would permit a model that uses higher levels, thus yielding higher performance, without requiring more parameters. Other promising directions include exploring applications of `ah-QCH` models for modeling behavior in practical settings such as markets or bargaining, and applying Bayesian analysis to additional behavioral models.

References

[1] Ayala Arad and Ariel Rubinstein. The 11-20 money request game: A level-k reasoning study. `http://www.tau.ac.il/~aradayal/moneyrequest.pdf`, May 2011.

[2] Colin F. Camerer. *Behavioral Game Theory: Experiments in Strategic Interaction.* Princeton University Press, 2003.

[3] Colin F. Camerer, Teck-Hua Ho, and Juin-Kuan Chong. A cognitive hierarchy model of games. *Quarterly Journal of Economics*, 119(3):861–898, 2004.

[4] David J. Cooper and John B. Van Huyck. Evidence on the equivalence of the strategic and extensive form representation of games. *Journal of Economic Theory*, 110(2):290–308, 2003.

[5] Miguel Costa-Gomes, Vincent P. Crawford, and Bruno Broseta. Cognition and behavior in normal-form games: an experimental study. Discussion paper 98-22, UCSD, 1998.

[6] Miguel Costa-Gomes, Vincent P. Crawford, and Bruno Broseta. Cognition and behavior in normal-form games: An experimental study. *Econometrica*, 69(5):1193–1235, 2001.

[7] Xi Alice Gao and Avi Pfeffer. Learning game representations from data using rationality constraints. In *UAI*, pages 185–192, 2010.

[8] Jeff Gill. *Bayesian methods: A social and behavioral sciences approach.* CRC press, 2002.

[9] Jacob K. Goeree and Charles A. Holt. Ten little treasures of game theory and ten intuitive contradictions. *American Economic Review*, 91(5):1402–1422, 2001.

[10] Ernan Haruvy and Dale O. Stahl. Equilibrium selection and bounded rationality in symmetric normal-form games. *Journal of Economic Behavior and Organization*, 62(1):98–119, 2007.

[11] Ernan Haruvy, Dale O. Stahl, and Paul W. Wilson. Modeling and testing for heterogeneity in observed strategic behavior. *Review of Economics and Statistics*, 83(1):146–157, 2001.

[12] Richard D. McKelvey and Thomas R. Palfrey. Quantal response equilibria for normal form games. *Games and Economic Behavior*, 10(1):6–38, 1995.

[13] Rosemarie Nagel. Unraveling in guessing games: An experimental study. *American Economic Review*, 85 (5):1313–1326, 1995.

[14] Radford M. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

[15] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7(4):308–313, 1965.

[16] Christian P. Robert and George Casella. *Monte Carlo statistical methods.* Springer Verlag, 2004.

[17] Brian W. Rogers, Thomas R. Palfrey, and Colin F. Camerer. Heterogeneous quantal response equilibrium and cognitive hierarchies. *Journal of Economic Theory*, 144(4):1440–1467, July 2009.

[18] Dale O. Stahl and Ernan Haruvy. Level-n bounded rationality and dominated strategies in normal-form games. *Journal of Economic Behavior and Organization*, 66(2):226–232, 2008.

[19] Dale O. Stahl and Paul W. Wilson. Experimental evidence on players' models of other players. *Journal of Economic Behavior and Organization*, 25(3):309–327, 1994.

[20] Dale O. Stahl and Paul W. Wilson. On players' models of other players: Theory and experimental evidence. *Games and Economic Behavior*, 10(1):218–254, 1995.

[21] Georg Weizsäcker. Ignoring the rationality of others: evidence from experimental normal-form games. *Games and Economic Behavior*, 44(1):145–171, 2003.

[22] James R. Wright and Kevin Leyton-Brown. Beyond equilibrium: Predicting human behavior in normal-form games. In *AAAI*, pages 901–907, 2010.

[23] Michael Wunder, Michael Kaisers, John Robert Yaros, and Michael Littman. Using iterated reasoning to predict opponent strategies. In *AAMAS*, pages 593–600, May 2011.

Session 5E
Game & Agent Theories

Scaling Simulation-Based Game Analysis through Deviation-Preserving Reduction

Bryce Wiedenbeck and Michael P. Wellman
University of Michigan
{btwied,wellman}@umich.edu

ABSTRACT

Multiagent simulation extends the reach of game-theoretic analysis to scenarios where payoff functions can be computed from implemented agent strategies. However this approach is limited by the exponential growth in game size relative to the number of agents. Player reductions allow us to construct games with a small number of players that approximate very large symmetric games. We introduce *deviation-preserving reduction*, which generalizes and improves on existing methods by combining sensitivity to unilateral deviation with granular subsampling of the profile space. We evaluate our method on several classes of random games and show that deviation-preserving reduction performs better than prior methods at approximating full-game equilibria.

Categories and Subject Descriptors

J.4 [**Social and Behavioral Sciences**]: Economics

General Terms

Algorithms, Economics

Keywords

empirical game theory, simulation-based game theory, game reduction

1. INTRODUCTION

Game-theoretic analysis plays an increasingly prominent role in research on understanding and designing multiagent systems. Agent-based simulation offers the potential to increase the scope of applicability for game theory, beyond those game scenarios that can be described straightforwardly and solved analytically. In the simulation-based approach, rather than directly express all payoffs for a game, the analyst describes an environment procedurally and then computes payoffs by simulation of agent interactions in that environment.

Simulation enables analysis of many rich strategic environments, but determining payoffs for a large game in this way may be prohibitively expensive. Straightforward estimation of a payoff function requires simulation of every possible combination, or *profile*, of agent strategies. If the environment is stochastic, then many simulation runs may be necessary to obtain a reasonable estimate of even a single profile. For multiagent interactions that extend over time, or are otherwise complex, the computational cost of simulation may severely limit the number of profiles—and therefore the size of the game—that can be considered in such an analysis.

We focus for most of this paper on *symmetric games*, in which all agents have the same set of available strategies and payoffs depend only on the number of agents playing each strategy, not on the specific identities of those agents. Formally, a symmetric game is a tuple $\Gamma = (N, S, u)$, where N is the number of agents, S is the set of strategies available to all agents, and the utility function $u(s, \vec{s})$ gives the payoff to any agent playing strategy s in profile $\vec{s}$. To conduct a complete analysis of Γ, we require that u specifies payoffs for all possible profiles. A symmetric game with N agents and $|S|$ strategies contains $\binom{N+|S|-1}{N}$ profiles.[1] For a sense of how great a burden this imposes, consider that a symmetric game with 15 agents and 15 strategies contains over 77 million profiles, so if estimating a profile's payoff through simulation required one second, constructing the full game would take more than two years.

We seek to combat this exponential growth using a technique broadly known as *player reduction*. Player reductions approximate games with many agents by constructing smaller games that aggregate over those agents in some way. Equilibria of the reduced game can then be viewed as approximate equilibria of the full game.

As an example, consider trading in continuous double auctions (CDAs)—a problem of agent strategy that has been extensively investigated through simulation. We review the coverage of several studies that employed simulation to estimate payoff functions for purposes of game-theoretic or evolutionary analysis. In the first empirical game analysis of CDA strategy, Walsh et al. [15] analyzed a 20-player game with three strategies. The 231 distinct profiles were within their simulation budget, whereas adding just one

[1]To see this, note that we can describe a profile in terms how many agents play each strategy. Suppose an ordering of strategies, and consider a representation that indicates players by one symbol (**.**) and partitions by another (**|**). For instance, with $S = \{s_1, s_2, s_3, s_4\}$ and $N = 6$, the profile **...|..||.** has three agents playing s_1, two s_2, and one s_4. The representation contains $N + |S| - 1$ total symbols, and the choice of which N of them to make players (or equivalently, choice of partitions) uniquely defines a profile.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

more strategy would have entailed estimating 1540 more. Vytelingum et al. [14] likewise considered a 20-player game, though one that imposed symmetry only within the subgroups of 10 buyers and 10 sellers. Their study also compared three strategies, but for tractability and to facilitate visualization of their evolutionary traces, they limited analysis to two strategies at a time, which requires 121 profiles for each strategy pair and scenario combination.[2] Phelps et al. [8] covered up to four strategies, in a 12-agent simulation employing another form of double auction mechanism (455 profiles). The study of Tesauro and Bredin [11] considered as many as 44 trading agents with three different strategies, but their analysis evaluated only profiles where agents were evenly divided across two strategies. This selection is similar to a two-player reduction according to the hierarchical method, as discussed below. Tesauro and Das [12] covered five strategies for a 20-agent scenario, this time with a mix of evenly-divided profiles and profiles where only one agent deviates from a homogeneous profile. As we see below, this is suggestive of the deviation-preserving reduction method we introduce here. In by far the most comprehensive CDA simulation study to date, Schvartzman and Wellman [10] systematically evaluated 14 strategies in a 16-agent scenario. This was rendered feasible only by virtue of their reduction to a four-player game, comprising 2380 profiles as opposed to 68 million in the unreduced game. Overall, we see that many-agent simulation studies either adopt player reductions, or make do with very narrow strategy exploration.

In this paper, we propose and study *deviation-preserving reduction*, which renders reduced-game equilibria more informative with respect to the full game. We start in Section 2 by reviewing existing methods for player reduction. Section 3 introduces deviation-preserving reduction, and explains how our new method is designed to combine the best aspects of its predecessors. Section 4 evaluates the reductions, and Section 5 shows how both our reduction and previous ones can be extended to games that are symmetric only with respect to a partition of players into roles.

2. BACKGROUND: PLAYER REDUCTION

Two methods for player reduction have been proposed in the literature: *hierarchical reduction* [16], and *twins reduction* [4]. Both methods are defined with respect to symmetric games. They define a subset of the profiles in the given (*full*) game, and map the payoffs of these profiles to a payoff function defined over a game with fewer players (the *reduced game*). Analyses of the reduced game are then interpreted as approximately applying to the full game.

2.1 Hierarchical Reduction

Of the two existing methods, hierarchical reduction is the more extensively used [1, 5, 10]. Hierarchical reduction works by grouping agents into coalitions that are constrained to act together. One player in the reduced game selects an action to be played by all agents in a coalition, and receives the payoff to any agent playing that strategy. To capture this formally, we introduce the following notation: a strategy profile $\vec{s} = \langle c_1 \times s_1, \ldots, c_{|S|} \times s_{|S|} \rangle$ of game Γ consists of strategies $s_i \in S$ and integer counts $c_i \geq 0$ for each strategy such that $\sum_{i=1}^{|S|} c_i = N$. When $c_i = 0$, we may omit it from the expression.

The hierarchical reduction of Γ to $n < N$ players is defined as $HR_n(\Gamma) = \left(n, S, u^{HR}\right)$, where

$$u^{HR}\left(s, \langle c_1 \times s_1, \ldots, c_{|S|} \times s_{|S|} \rangle\right) = u\left(s, \left\langle \frac{N}{n} c_1 \times s_1, \ldots, \frac{N}{n} c_{|S|} \times s_{|S|} \right\rangle\right).$$

This definition follows previous applications of hierarchical reduction in assuming that N is an integer multiple of n. In our evaluation we employ a generalized version (described in Section 4.1) that allows reduction to numbers of players that do not evenly divide the number of agents in the full game.

For illustration, consider a full game with $N = 25$ agents, and a hierarchical reduction to $n = 5$ players. The action of each reduced-game player is played by five agents in the full game, so the reduced-game profile $\langle 2 \times s_1, 1 \times s_2, 2 \times s_3 \rangle$ corresponds to the full-game profile $\langle 10 \times s_1, 5 \times s_2, 10 \times s_3 \rangle$.

The main idea behind hierarchical reduction is that though the payoff to a particular strategy generally varies with the number of agents that play each strategy, it often can be expected to do so smoothly. Kearns and Mansour [6] formalize a related condition called *bounded influence* to define a class of compactly representable and solvable games. Whereas it is easy to construct games that violate this assumption, in many natural symmetric games, the payoffs are smooth in this way.

However, $HR_n(\Gamma)$ lacks crucial information relevant to Nash equilibria of Γ. In a Nash equilibrium of Γ, no individual agent can gain by deviating to another strategy, but the hierarchical reduction contains no information about unilateral deviations. In an equilibrium of $HR_n(\Gamma)$, no $\frac{N}{n}$-agent coalition can gain by all deviating to the same strategy, but there are many cases in which these conditions differ substantially.

Consider a network formation game [3] in which agents create links to one another. Agents gain from being in a connected network, but incur a cost for each link they create. In such a game we can envision a full-game equilibrium that is not an equilibrium of the reduced game, as well as a reduced-game equilibrium that is not an equilibrium of the full game. If network effects are large, but cost of creating links is high, there could be an equilibrium of the full game where agents create no links. However, if several players are allowed to deviate together they may create a sufficiently dense network to overcome the link-creation cost: this would be a beneficial deviation in the reduced game, so the full-game equilibrium would not be found. Under different parameters, there may be a spurious equilibrium in the reduced game where all players contribute links to the network, and no player can gain by deviating because if all agents represented by one reduced-game player changed strategies simultaneously, the network would collapse. On the other hand, a unilaterally deviating agent in the full game might have a much smaller impact and still receive the network benefits while avoiding the link creation cost.

2.2 Twins Reduction

The natural solution to this problem is to incorporate information about the value of unilateral agent deviations into the payoffs of the reduced game. Ficici et al. [4] propose a method called *twins reduction* that takes a first step in this

[2]These same authors in earlier work [13] evaluated a 20-player three-strategy CDA game.

direction. The twins reduction of a symmetric game[3] is a 2-player game, $TR(\Gamma) = \left(2, S, u^{TR}\right)$, where each player views itself as controlling one agent in the full game, and the opponent as controlling all remaining agents:

$$u^{TR}\left(s, \langle 1\times s, 1\times s'\rangle\right) = u\left(s, \langle 1\times s, (N-1)\times s'\rangle\right).$$

Note that the payoffs for the two strategies in a twins reduction profile $\langle 1\times s, 1\times s'\rangle$, $s \neq s'$, correspond to two different profiles in the full game:

- $\langle 1\times s, (N-1)\times s'\rangle$ and
- $\langle (N-1)\times s, 1\times s'\rangle$,

but that the reduced game is still symmetric.

Ficici et al. [4] advocate constructing twins reduction games not by explicitly simulating these full-game profiles, but by sampling random profiles from the full game and determining the payoffs by linear regression on the number of agents playing each strategy. We refer to this approach as *TR-R*, where the second "R" stands for "regression". We consider the direct simulation approach a more appropriate benchmark, but evaluate both methods in our experiments.

The advantage of the twins reduction is that it captures information about individual agents' incentives to deviate. Its major disadvantage is that it is limited to two players, and can therefore give only an extremely coarse-grained view of the game. In general, a reduced-game representation will have difficulty capturing equilibria of the full game that have support size (number of distinct strategies played with positive probability) larger than n, the reduced number of players, as no profiles of the reduced game capture the interaction of all strategies in the support set. Since the twins reduction ($n = 2$) never contains profiles where more than two strategies are played, it is particularly restrained by this limitation.

3. DEVIATION-PRESERVING REDUCTION

We propose a new game reduction method that combines the sensitivity to unilateral deviation afforded by twins reduction with the profile-space granularity of hierarchical reduction. We call this method *deviation-preserving reduction.* In a deviation-preserving reduction game, each player views itself as controlling a single agent in the full game, but views the profile of opponent strategies in the reduced game as an aggregation of all other agents in the full game. Formally, $DPR_n(\Gamma) = \left(n, S, u^{DPR}\right)$, where

$$u^{DPR}\left(s, \langle c_1\times s_1, \ldots, c_s\times s, \ldots\rangle\right) =$$
$$u\left(s, \left\langle \frac{N-1}{n-1}c_1\times s_1, \ldots, \left[\frac{N-1}{n-1}(c_s-1)+1\right]\times s, \ldots\right\rangle\right).$$

In a hierarchical reduction, the proportion of agents playing each strategy is the same in the full and reduced games. Under deviation-preserving reduction, analogously, the proportion of *opponents* playing a strategy in the full and reduced games is the same from each player's perspective. And as in a twins reduction, each player in a deviation-preserving

[3]The original definition [4] applies to a somewhat broader class: role-symmetric games with identical strategies. In Section 5 we describe how to extend player reductions, including twins reduction, to the entire class of role-symmetric games.

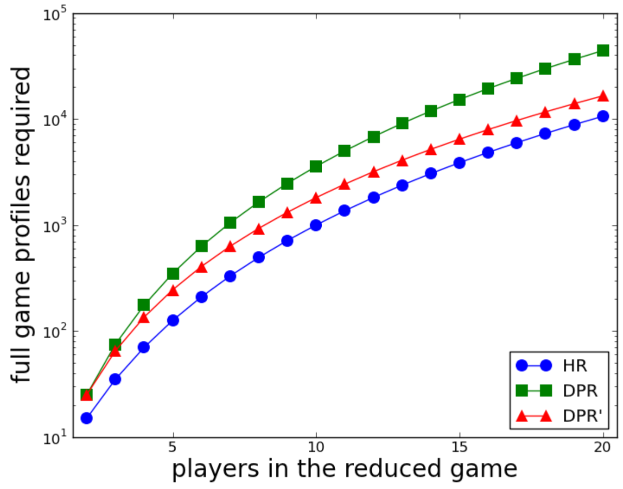


Figure 1: Number of full-game profiles required to construct reduced games (log scale), for $|S| = 5$.

reduction game is sensitive to the payoffs of exactly one agent in the full game. As a consequence of this sensitivity to single agents, the deviation-preserving reduction game can identify exact symmetric pure strategy equilibria of the full game if they exist.

PROPOSITION 1. *A profile $\langle n\times s\rangle$ is a Nash equilibrium of $DPR_n(\Gamma)$ if and only if the profile $\langle N\times s\rangle$ is a Nash equilibrium of Γ.*

PROOF. The profile $\langle n\times s\rangle$ is a NE when $u^{DPR}\left(\langle n\times s\rangle\right) \geq u^{DPR}\left(s, \langle (n-1)\times s, 1\times s'\rangle\right)$ for all $s' \in S$. This is the case exactly when $u\left(s, \langle N\times s\rangle\right) \geq u\left(s, \langle (N-1)\times s, 1\times s'\rangle\right)$ for all $s' \in S$. $\square$

This property also holds for twins reduction games, because the deviation preserving reduction is a strict generalization of the directed-sampling twins reduction: $TR(\Gamma) = DPR_2(\Gamma)$.

To construct each profile's payoffs in a deviation-preserving reduction game, several profiles from the full game must be simulated. Returning to the example of a 25-agent full game and a 5-player reduced game, the profile $\langle 2\times s_1, 1\times s_2, 2\times s_3\rangle$ in the deviation-preserving reduction game employs payoff values from several profiles in the full game:

$$u^{DPR}\left(s_1, \langle 2\times s_1, 1\times s_2, 2\times s_3\rangle\right) = u(s_1, \langle 7\times s_1, 6\times s_2, 12\times s_3\rangle)$$
$$u^{DPR}\left(s_2, \langle 2\times s_1, 1\times s_2, 2\times s_3\rangle\right) = u(s_2, \langle 12\times s_1, 1\times s_2, 12\times s_3\rangle)$$
$$u^{DPR}\left(s_3, \langle 2\times s_1, 1\times s_2, 2\times s_3\rangle\right) = u(s_3, \langle 12\times s_1, 6\times s_2, 7\times s_3\rangle)$$

Note that we again assume divisibility: in this case, $n-1$ has to divide $N-1$ for the aggregation of opponents to be precise. As with hierarchical reduction, we can extend the definition to reduced games with any number of players, as described in Section 4.1. We quantify the number of profile simulations required for deviation-preserving reduction in the following proposition.

PROPOSITION 2. *Constructing $DPR_n(\Gamma = (N, S, u))$ requires simulating $|S|\binom{n+|S|-2}{n-1}$ full-game profiles.*

PROOF. In each profile of the deviation-preserving reduction game, $n-1$ of the players each control $\frac{N-1}{n-1}$ full-game agents. The set of all such profiles can be viewed as an $(n-1)$-player symmetric game, so we know that there are $\binom{n+|S|-2}{n-1}$ of them. Each of these profiles must be paired with each $s \in S$, so $|S|\binom{n+|S|-2}{n-1}$ profiles must be simulated. $\square$

Proposition 2 shows that constructing $DPR_n(\Gamma)$ requires simulating strictly more profiles than $HR_n(\Gamma)$, but by a factor of at most $|S|$. As we show in Section 4, the extra profiles comprising this constant factor can contribute to significantly improved accuracy. Even so, we would like to minimize the number of simulations required when possible, and therefore also consider a variant of the deviation-preserving reduction, which we call DPR'.

The idea behind DPR' is that many of the profiles simulated to construct a deviation-preserving reduction game are quite similar. For example, in the 5-player deviation-preserving reduction of the 25-agent game, the payoff to strategy s_1 in the full-game profile $\vec{s}_a = \langle 7\times s_1, 6\times s_2, 12\times s_3\rangle$ is employed in the reduced-game profile $\langle 2\times s_1, 1\times s_2, 2\times s_3\rangle$. The payoff for s_2 in reduced-game profile $\langle 1\times s_1, 2\times s_2, 2\times s_3\rangle$ is derived from $\vec{s}_b = \langle 6\times s_1, 7\times s_2, 12\times s_3\rangle$, which differs from $\vec{s}_a$ only in that a single agent has switched from s_1 to s_2, both of which are played by many other agents. If we believe our assumption—inherited from hierarchical reduction—that payoffs vary smoothly in the number of agents playing each strategy, we should expect the payoffs to strategy s_1 in profiles $\vec{s}_a$ and $\vec{s}_b$ to be very similar (likewise for s_2), suggesting that we could get away with simulating only one of the two.

Formally, $DPR'(\Gamma) = (n, S, u^{DPR'})$, where $u^{DPR'}$ is defined as follows. Let $\vec{s} = \langle c_{\min}\times s_{\min}, \ldots, c_s\times s, \ldots\rangle$, where $s_{\min}$ is the first strategy played by at least one agent. If $c_s = 1$ or $c_s = c_{\min}$, then $u^{DPR'}(s,\vec{s}) = u^{DPR}(s,\vec{s})$. Otherwise, $u^{DPR'}(s,\vec{s})$ is given by

$$u(s, \left\langle \left[\frac{N-1}{n-1}c_{\min}+1\right]\times s_{\min}, \ldots, \frac{N-1}{n-1}(c_s-1)\times s, \ldots \right\rangle).$$

The result is that when DPR would prescribe simulation of several profiles that differ only by deviation of a single agent (and no strategy is played by only one agent), DPR' requires that only one be simulated. From among these profiles, DPR' selects the one in which the lowest-numbered strategy by which they differ is played most. In the example above, payoffs

- $u^{DPR'}(s_1, \langle 2\times s_1, 1\times s_2, 2\times s_3\rangle)$ and
- $u^{DPR'}(s_2, \langle 1\times s_1, 2\times s_2, 2\times s_3\rangle)$

both come from full-game profile $\langle 7\times s_1, 6\times s_2, 12\times s_3\rangle$. The savings in terms of profiles sampled are illustrated in Figure 1. In that graph, the curve for DPR' follows the formula $|\Gamma| = |S|\binom{n+|S|-2}{n-1} - (n-2)\binom{n+|S|-3}{n-1}$. DPR' always requires more full-game profiles than HR, and fewer than DPR, except when $n=2$, where both are equivalent to TR.

4. EMPIRICAL EVALUATION

The goal of a player reduction is to replace a full game that is too large to effectively analyze with a more manageable reduced game. To compare reduction methods, we therefore need to evaluate how well analysis performed on a reduced game translates back to the full game. This presents a problem for evaluation, in that full games of interest are too big to effectively analyze. For example, in the simulated credit network games discussed below, we construct 12-agent, 6-strategy full games; we would like to analyze reductions of 60-agent games, but even with just six strategies, the full game would consist of 8,259,888 profiles. We therefore compromise by reducing several types of medium-sized games to very small ones. If one reduction consistently performs better in such cases, we take it as an indication that the same will hold for reductions of very large games.

4.1 Regret of Reduced Game Equilibria

Numerous methods for analyzing games exist, but the most important is finding Nash equilibria. Because player reductions work with symmetric games, we evaluate them primarily by how well symmetric mixed strategy Nash equilibria computed in the reduced game approximate symmetric mixed strategy equilibria of the full game. Our primary measure for the quality of reduced-game equilibria is *regret.* The regret $\epsilon(\vec{\sigma})$ of a symmetric mixed strategy profile $\vec{\sigma}$, in which all players play mixed strategy σ is the maximum gain any player could achieve by deviating to a pure strategy:

$$\epsilon(\vec{\sigma}) = \max_{s\in S} u(s, \vec{\sigma}_{-i}),$$

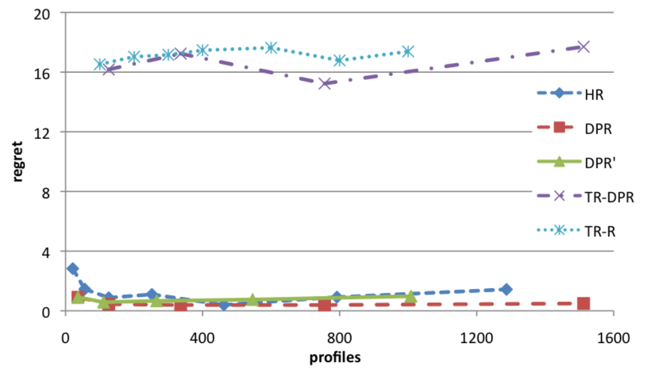
where $u(s,\vec{\sigma}_{-i})$ is the expected payoff to a player playing s when all others play σ. A Nash equilibrium has zero regret, but a symmetric mixed profile $\vec{\sigma}$ that is an equilibrium of the reduced game will generally have have positive regret with respect to the full game. Such a $\vec{\sigma}$ can be viewed as an *approximate*, or $\epsilon(\vec{\sigma})$-Nash equilibrium of the full game, where the lower the regret, the better the approximation.

However, we cannot simply compare the regret of equilibria from k-player reduced games under each method. The first problem is that the number of players in a twins reduction game is not scalable. Moreover, since the goal of player reduction is to simulate fewer profiles, the relevant comparison is not the number of players in the reduced game, but the number of profiles required to construct it, and DPR_k always requires sampling strictly more profiles than HR_k. For example, in the 12-agent 6-strategy game instances below, $|DPR_3| = 126 = |HR_4|$. Because the directed-simulation twins reduction always requires $\frac{|S|^2(|S|-1)}{2}$ profiles, we compare to TR by addressing the question of whether lower regret can be achieved by a method that samples more profiles. Twins reduction with regression can use any set of profiles; in our experiments we varied the size of the set over a range similar to that required to construct the various reduced games.

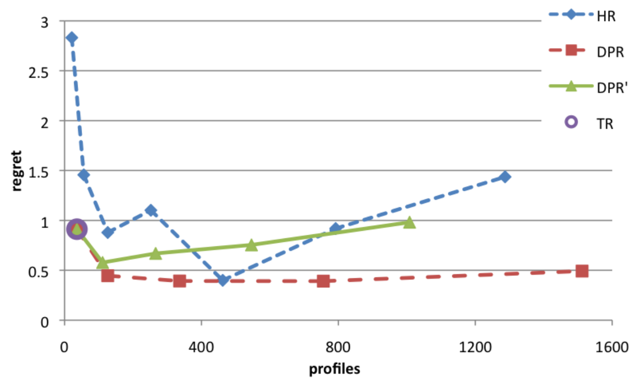
Hierarchical reduction as defined by Wellman et al. [16] requires that the number of reduced-game players n divide the number of full-game agents N. By analogy, in our definition of deviation preserving reduction above, we assume that $n-1$ divides $N-1$. In our experiments, we perform reductions of both varieties where these conditions do not hold. We extend the definition of HR_n to allow indivisibility as follows.

$$u^{HR}(s, \langle c_1\times s_1, \ldots, c_{|S|}\times s_{|S|}\rangle) = \\ u(s, \left\langle \lfloor \frac{N}{n}c_1+1\rfloor\times s_1, \ldots, \lfloor\frac{N}{n}c_{j+1}\rfloor\times s_{j+1}, \ldots\right\rangle),$$

where $j = N - \sum_i \lfloor\frac{N}{n}c_i\rfloor$. That is, the number of opponents



(a) all reduction types



(b) rescaled to exclude regression-based twins reductions

Figure 2: Average full-game regret of reduced-game equilibria in local effect games. $N = 12$, $|S| = 6$, $2 \leq n \leq 8$.

playing strategy s_i is the integral part of $\frac{N}{n}c_i$, with extra player slots allocated one each to strategies with lower indices. For example, when we construct HR_5 of a game with 12 players, the reduced-game profile $\langle 1\times s_1, 3\times s_2, 1\times s_3\rangle$ corresponds to the full-game profile $\langle 3\times s_1, 7\times s_2, 2\times s_3\rangle$. We extend the definition of DPR_n to handle indivisibility in a similar manner.

We evaluate the reductions using two classes of random games: congestion games [9], in which agents select a fixed-size subset of available facilities and payoffs are decreasing in the number of agents choosing a facility; and local effect games [7], which have a graph over actions and each action's payoff is a function of the number of agents choosing it and adjacent actions. We randomly varied the payoff function parameters of these games to create 250 game instances for each test described below. We also evaluate on one simulated game class, based on a scenario of *credit network formation* [2]. In the model of credit networks employed in this scenario, directed links represent credit issued to other agents: agents wish to transact with one-another, but issuing credit bears risk in that debtors may default. In the credit network formation game, payoffs are determined by simulating a sequence of transactions and defaults on the network induced by agent strategies. We sampled each profile of a 12-agent, 6-strategy credit network game 100 times, and randomly recombined these samples to create 250 game instances.

The first finding of note is that twins reduction performs very poorly with linear regression. The top line in Figure 2a shows the regret of equilibria found in *TR-R* games with random sampling of profiles, which is an order of magnitude worse than the *HR*, *TR*, *DPR*, and DPR'. Two observations led us to try the method labeled *TR–DPR*: first, that sampling profiles according to uniform agent play leads to a very low likelihood of observing payoffs for profiles where most agents play the same strategy, and these are exactly the profiles whose payoffs the regression estimates. Second, simulating all the profiles for *DPR* or DPR' makes available a substantial amount of payoff data that goes unused in constructing the reduced game. We therefore thought to try using all of the profiles simulated for the deviation-preserving reduction as input to the linear regression of *TR-R*. As is clear from Figure 2a, this improves very little on random sampling.

In retrospect, it is not particularly surprising that approximating payoffs by linear regression performs so poorly: all of our example games and most games requiring simulation have nonlinear payoffs. A better regression model could potentially alleviate this problem, but choosing one requires knowledge of the game's payoff function that may not be available when payoffs are determined by simulation. We also ran *TR-R* and *TR–DPR* on each of the other game classes, but the results are similarly poor, and are excluded from subsequent figures.

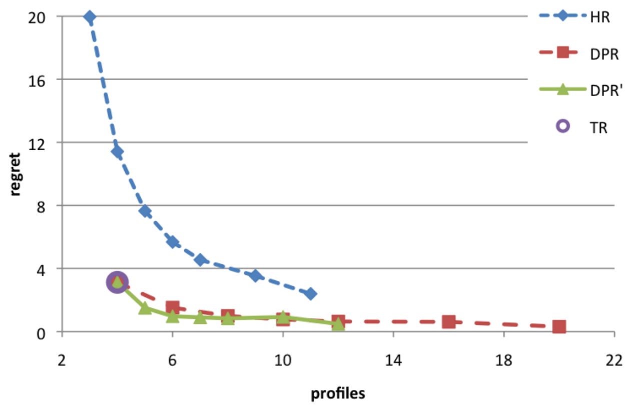


Figure 3: Full-game regret of reduced-game equilibria in congestion games. $N = 100$, $|S| = 2$, $2 \leq n \leq 10$.

Figures 2b, 3, and 4 show that deviation-preserving reduction outperforms hierarchical reduction and twins reduction in a wide variety of settings. In 12-agent, 6-strategy local effect games, *DPR* is clearly better than *HR*, but the comparison to DPR' is less conclusive. We were surprised to find that hierarchical reduction would perform worse with increased reduced-game size, which corresponds to increased abstraction granularity. We note, however, that the 5, 7, and 8-player reduced games where *HR* performs poorly are exactly the cases where n does not divide $N = 12$. This also leads us to observe that because 11 is prime, the deviation-preserving reduction never has the advantage $n - 1$ dividing $N - 1$, and yet consistently performs well. The results from 12-agent, 6-strategy congestion games (not shown) are broadly similar.

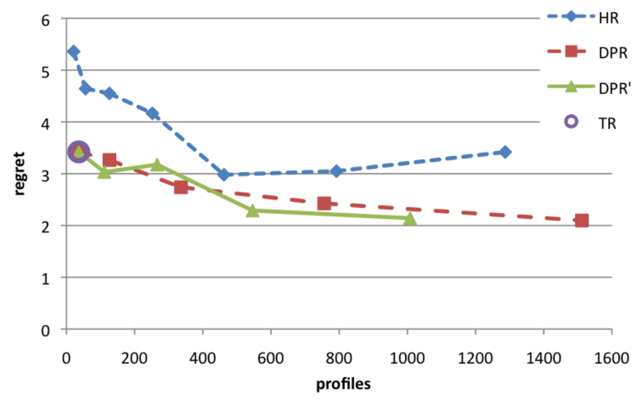


Figure 4: Average full-game regret of reduced-game equilibria in credit network games. $N = 12$, $|S| = 6$, $2 \leq n \leq 8$.

In an attempt to get at the effect of very substantial player reductions, we created 100-agent, 2-strategy congestion games. The results in Figure 3 show clear separation between hierarchical reduction and both variants of deviation-preserving reduction, suggesting that as the number of players grows, the relative difference between DPR and DPR' may be smaller. Results for the 12-agent, 6-strategy credit network game appear in Figure 4. Here again, DPR and DPR' perform similarly, and better than HR.

Across all game classes and sizes examined (including those not shown) deviation-preserving reduction of any given size outperforms the hierarchical reduction with at least as many profiles that is closest in size. This means that for any size hierarchical reduction, there exists a better deviation-preserving reduction that requires simulating fewer profiles. Virtually all of these differences are significant at $p < 0.05$; the only exceptions are 4-player DPR versus 6-player HR in the 12-player congestion game (Figure 3) and 12-player credit network game (Figure 4). In addition, DPR_3 outperforms TR across all game classes; the difference is significant at $p < 0.05$ in all cases except the credit network game. The difference between DPR_4 and TR is significant in all cases.

4.2 Comparison to Full-Game Equilibria

We also compared reduced-game equilibria under HR and DPR to equilibria from 12-player, 6-strategy full games using two metrics: similarity of support sets, and L^2 distance between distributions. Table 1 shows the number of strategies by which the support sets of full and reduced-game equilibria differ. Here, we consider a strategy to be in the support of a symmetric ϵ-Nash equilibrium if it is played with probability 0.01 or greater. In nearly all cases support sets of DPR_n match match those of full-game equilibria significantly ($p < 0.05$) better than both HR_n and HR_{n+1}. In addition, for congestion games and local effect games, $DPR_{>2}$ significantly outperforms TR, whereas in credit network games, there is no significant difference between TR and DPR.

Table 2 presents a similar message, but in terms of the L^2 distances between the mixed strategy distributions in full and reduced-game equilibria. Again, DPR is significantly better than HR and TR for congestion and local effect games, while performing similarly on credit network games. As in Section 4.1, in these experiments, we compute

	credit network		congestion		local effect	
n	HR	DPR	HR	DPR	HR	DPR
2	3.72	1.49†	1.68	0.39†	2.72	0.45†
3	3.72	1.64†	0.99*	0.17†*	1.04*	0.20†*
4	3.72	1.60†	1.05	0.10†*	0.98	0.12†*
5	1.98*	1.54†	1.10	0.08†	1.01	0.09†
6	1.19†*	1.39	0.98	0.05†	0.85*	0.08†

Table 1: Reduced versus full-game NE support set difference. * indicates significant difference between n and $n-1$; † indicates significant difference between HR_n and DPR_n.

	credit network		congestion		local effect	
n	HR	DPR	HR	DPR	HR	DPR
2	0.713	0.703	0.435	0.069†	0.503	0.128†
3	0.764*	0.690†	0.141*	0.039†*	0.154*	0.064†*
4	0.860	0.640†	0.117*	0.022†*	0.117*	0.037†*
5	0.643*	0.641	0.141	0.019†*	0.144	0.027†*
6	0.467†*	0.564*	0.099*	0.018†	0.088*	0.026†

Table 2: Reduced versus full-game NE distribution L^2 distance. * indicates significant difference between n and $n-1$; † indicates significant difference between HR_n and DPR_n.

one symmetric mixed-strategy Nash equilibrium per game by running replicator dynamics initialized to the uniform mixture.

4.3 Dominated Strategies

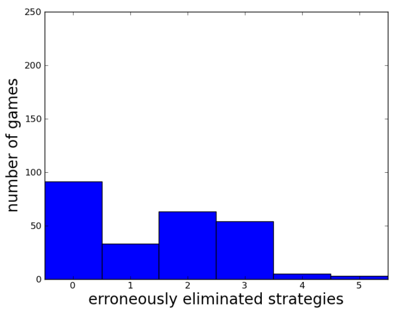
Another useful operation in the analysis of simulation-based games is to check for dominated strategies. A dominated strategy is one that no agent should ever play because there is an alternative strategy that is at least as good in response to any profile of opponent strategies. We ran experiments on 12-agent, 6-strategy congestion and credit network games (250 each), comparing the set of strategies that remain after iterated elimination of strictly dominated strategies in the full game against those that remain in 2, 4, and 6-player reduced games. We observed that DPR and DPR' produced very similar results, and that both improved over hierarchical and twins reduction. Figures 5 and 6 show histograms of the number of strategies eliminated in reduced games but not eliminated in full games.

In congestion games (Figure 5), twins reduction and both forms of deviation-preserving outperform hierarchical reduction, eliminating fewer strategies in the reduced game that survive in the full game, even when hierarchical reduction samples vastly more profiles. These congestion games often exhibit dominated strategies in the full game, but we almost never observed strategies surviving in reduced games that are dominated in the full game.

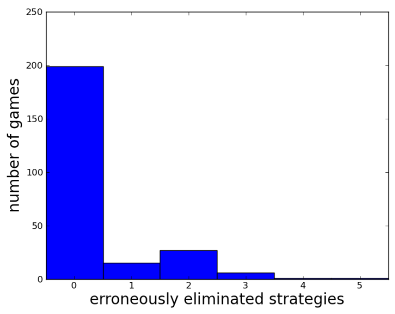
In credit network games (Figure 6), no strategies are dominated in the full game, but in the twins reduction game, many strategies are eliminated. Moving to DPR_4 or DPR'_4 solves this problem almost entirely. These experiments also confirm that for all reduction types, increasing the number of players in the reduced game reduces the number of strategies erroneously found to be dominated.

5. ROLE-SYMMETRIC GAMES

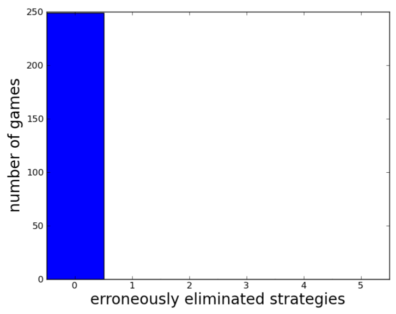
We can smoothly relax the constraint that games be fully



(a) HR_2 (21 profiles)

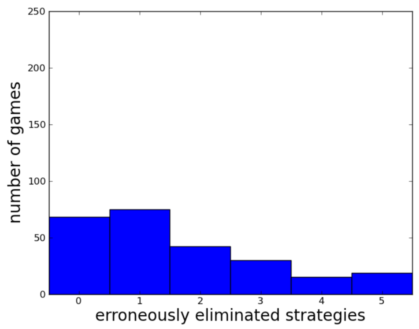


(b) HR_4 (126 profiles)

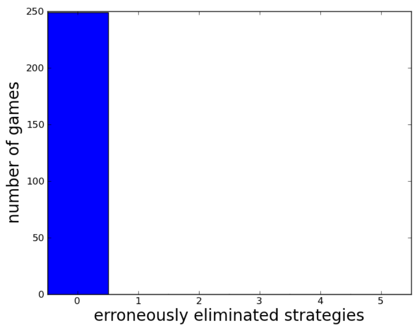


(c) $TR \equiv DPR_2 \equiv DPR'_2$ (36 profiles)

Figure 5: Histograms showing the number of strategies surviving iterated elimination of dominated strategies in full but not reduced congestion games. $N = 12$, $|S| = 6$, 250 random games. $TR \equiv DPR_2$ outperforms HR, sampling far fewer profiles.



(a) $TR \equiv DPR_2 \equiv DPR'_2$ (36 profiles)



(b) DPR'_4 (336 profiles)

Figure 6: Histograms showing the number of strategies surviving iterated elimination of dominated strategies in full but not reduced credit network games. $N = 12$, $|S| = 6$, 250 sample games. DPR'_4 avoids the aggressive elimination occurring in TR.

symmetric by assigning agents to roles, and enforcing symmetry only within these roles. Across roles, agents' strategy sets and payoffs can differ, but within a role, they are symmetric. Formally, a *role-symmetric game* is a tuple $\Gamma = (\{N_i\}, \{S_i\}, u)$, where the number of agents with role i is N_i, and agents with role i have strategy set S_i. Role-symmetric games provide a natural model for many settings where agents can be partitioned into meaningful categories, such as buyers and sellers in a market, or attackers and defenders in a security game. Role symmetry imposes no loss of generality on normal-form games, spanning the spectrum from complete asymmetry (each player has its own role) to full symmetry (a single role for everyone).

All of the player reduction methods discussed here can be straightforwardly extended to role-symmetric games. Consider for example the 20-agent continuous double auction study of Vytelingum et al. [14] with $N_1 = 10$ buyers and $N_2 = 10$ sellers. Instead of choosing n, the number of players in the reduced game, we must choose each $\{n_i\}$, the number of players with each role in the reduced game.

To perform a hierarchical reduction, a natural choice would be $n_1 = n_2 = 2$. This would involve simulating all profiles where 0, 5, or 10 agents play each buyer strategy, and a multiple of five agents likewise play each seller strategy.

With twins reduction, there are two players per role. Each player views itself as controlling a single agent, and the other player with the same role as controlling nine agents. It views the two other-role players as each representing half the ten agents with that role, so in the reduced-game profile

$$\langle 1\times s_{1.1}, 1\times s_{1.2}, 1\times s_{2.1}, 1\times s_{2.2}\rangle\,,$$

the payoff to buyer 1, who plays $s_{1.1}$, comes from full-game profile

$$\langle 1\times s_{1.1}, 9\times s_{1.2}, 5\times s_{2.1}, 5\times s_{2.2}\rangle\,.$$

The deviation-preserving reduction extends the twins reduction to more than two reduced-game players per role, maintaining the view that a reduced-game player controls a single agent, while the other players with the same role aggregate over *the rest* of the agents with that role, and players with another role aggregate over *all* agents with their role. With either hierarchical reduction or deviation-preserving reduction, it would be possible to choose $n_i \neq n_j$ if different granularity of reduction were desired for different roles.

This extension to role-symmetric games encompasses the broader class over which Ficici et al. [4] define the twins reduction. The clustering method by which they aggregate agents induces a role-symmetric game that restricts all roles to have the same strategy set (but allows different payoffs). They mention but do not develop the idea that the twins reduction might extend to role-symmetric games. To our knowledge, hierarchical reduction has not been applied to role-symmetric games.

6. CONCLUSIONS

Our new player reduction method, deviation-preserving reduction, combines the most appealing aspects of hierarchical reduction and twins reduction. It also performs better than both prior methods experimentally: equilibria from *DPR* games have lower full-game regret and more closely resemble full-game equilibria, even when sampling fewer full-game profiles. In addition, performing iterated elimination of dominated strategies on deviation-preserving reduction games stays more faithful to the full game compared to other player reductions. Our alternative DPR' formulation performs reasonably well in the same tests. The simulation savings from DPR' are greatest when the reduced game has many players but few strategies, so DPR' may prove useful in such cases. Though it may not be obvious how to choose between *DPR* and DPR', the evidence is quite compelling that deviation-preserving reduction is the best available player reduction method for analyzing large simulation-based games.

7. REFERENCES

[1] B.-A. Cassell and M. P. Wellman. Agent-based analysis of asset pricing under ambiguous information. In *SpringSim Agent-Directed Simulation Symposium*, 2011.

[2] P. Dandekar, A. Goel, M. P. Wellman, and B. Wiedenbeck. Strategic formation of credit networks. In *21st International Conference on World Wide Web*, Lyon, France, 2012.

[3] A. Fabrikant, A. Luthra, E. N. Maneva, C. H. Papadimitriou, and S. Shenker. On a network creation game. In *22nd ACM Symposium on Principles of Distributed Computing*, pages 347–351, 2003.

[4] S. G. Ficici, D. C. Parkes, and A. Pfeffer. Learning and solving many-player games through a cluster-based representation. In *24th Conference on Uncertainty in Artificial Intelligence*, pages 187–195, Helsinki, 2008.

[5] P. R. Jordan, C. Kiekintveld, and M. P. Wellman. Empirical game-theoretic analysis of the TAC supply chain game. In *6th International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1188–1195, Honolulu, 2007.

[6] M. Kearns and Y. Mansour. Efficient Nash computation in large population games with bounded influence. In *18th Conference on Uncertainty in Artificial Intelligence*, pages 259–266, Edmonton, 2002.

[7] K. Leyton-Brown and M. Tennenholtz. Local-effect games. In *18th International Joint Conference on Artificial Intelligence*, pages 772–780, Acapulco, 2003.

[8] S. Phelps, M. Marcinkiewicz, S. Parsons, and P. McBurney. A novel method for automatic strategy acquisition in n-player non-zero-sum games. In *5th International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 705–712, Hakodate, 2006.

[9] R. W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

[10] L. J. Schvartzman and M. P. Wellman. Stronger CDA strategies through empirical game-theoretic analysis and reinforcement learning. In *8th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 249–256, Budapest, 2009.

[11] G. Tesauro and J. L. Bredin. Strategic sequential bidding in auctions using dynamic programming. In *1st International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 591–598, Bologna, 2002.

[12] G. Tesauro and R. Das. High-performance bidding agents for the continuous double auction. In *3rd ACM Conference on Electronic Commerce*, pages 206–209, Tampa, 2001.

[13] P. Vytelingum, D. Cliff, and N. R. Jennings. Evolutionary stability of behavioural types in the continuous double auction. In *AAMAS-06 Joint Workshop on Trading Agent Design and Analysis and Agent Mediated Electronic Commerce*, Hakodate, 2006.

[14] P. Vytelingum, D. Cliff, and N. R. Jennings. Strategic bidding in continuous double auctions. *Artificial Intelligence*, 172:1700–1729, 2008.

[15] W. E. Walsh, R. Das, G. Tesauro, and J. O. Kephart. Analyzing complex strategic interactions in multi-agent systems. In *AAAI-02 Workshop on Game-Theoretic and Decision-Theoretic Agents*, Edmonton, 2002.

[16] M. P. Wellman, D. M. Reeves, K. M. Lochner, S.-F. Cheng, and R. Suri. Approximate strategic reasoning through hierarchical reduction of large symmetric games. In *20th National Conference on Artificial Intelligence*, pages 502–508, Pittsburgh, 2005.

Towards Tractable Boolean Games

Paul E. Dunne
Department of Computer Science
Uinversity of Liverpool
Liverpool L69 3BX, UK
ped@csc.liv.ac.uk

Michael Wooldridge
Department of Computer Science
Uinversity of Liverpool
Liverpool L69 3BX, UK
mjw@csc.liv.ac.uk

ABSTRACT

Boolean games are a compact and expressive class of games, based on propositional logic. However, Boolean games are computationally complex: checking for the existence of pure Nash equilibria in Boolean games is Σ_2^p-complete, and it is co-NP-complete to check whether a given outcome for a Boolean game is a pure Nash equilibrium. In this paper, we consider two possible avenues to tractability in Boolean games. First, we consider the development of alternative solution concepts for Boolean games. We introduce the notion of k-bounded Nash equilibrium, meaning that no agent can benefit from deviation by altering fewer than k variables. After motivating and discussing this notion of equilibrium, we give a logical characterisation of a class of Boolean games for which k-bounded equilibria correspond to Nash equilibria. That is, we precisely characterise a class of Boolean games for which all Nash equilibria are in fact k-bounded Nash equilibria. Second, we consider classes of Boolean games for which computational problems related to Nash equilibria are easier than in the general setting. We first identify some restrictions on games that make checking for beneficial deviations by individual players computationally tractable, and then show that certain types of *socially desirable* equilibria can be hard to compute even when the standard decision problems for Boolean games are easy. We conclude with a discussion of related work and possible future work.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; I.2.4 [**Knowledge representation formalisms and methods**]

General Terms

theory

Keywords

Boolean games, complexity, game theory, logic, Nash equilibria

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

1. INTRODUCTION

Game-theoretic solution concepts such as Nash equilibria were originally formulated independently of considerations of whether or how they might be practically computed. Since solution concepts typically attempt to capture a notion of optimal choice in strategic settings, it is therefore not at all surprising that solution concepts are hard to compute in practice for many natural and important classes of games (see, e.g., [3, 12, 4, 15, 8]). The problem of classifying exactly the complexity of solution concepts in various settings has been an area of significant research activity over the past two decades. Most notably, the problem of classifying the complexity of computing mixed strategy Nash equilibria in 2-person strategic form games turned out to be one of the major challenges in complexity theory in the first decade of the 21st century [4].

Given that solution concepts are very often computationally complex, at least two possible routes to tractability suggest themselves:

- First, we can try to identify useful classes of games or representations of games for which solution concepts can be easily computed. In cooperative game theory, for example, the marginal contribution net representation allows for the efficient computation of the Shapley value solution concept [10].
- Second, we can develop alternative solution concepts, which lend themselves to efficient computation. For example, the notion of ϵ-Nash equilibrium has been developed, which relaxes the strict notion of Nash equilibrium by requiring that no player can gain more than ϵ in a deviation from a given strategy profile [13, p.83].

Our aim in the present paper is to consider these possibilities in the context of *Boolean games* [9, 2, 6, 7]. Boolean games are a simple, compact, and expressive class of games based on propositional logic. In a Boolean game, each player i has under its unique control a set of Boolean variables Φ_i, drawn from an overall set of Boolean variables Φ. Player i is at liberty to assign values to these variables as it chooses. The strategies or choices available to i correspond to all possible Boolean assignments that can be made to these variables. The outcome of a Boolean game is a valuation for the variables Φ, which will be composed from the individual assignments made by the players in the game to their variables. In addition, each player i has a goal that it desires to be achieved: the goal is represented as a Boolean formula γ_i, and this goal formula may contain variables under the control of other players. A player is satisfied with an overall outcome if that outcome satisfies its goal γ_i, and is unsatisfied otherwise. The fact that the achievement of one agent's goal may depend on the choices of other agents is what gives Boolean games their strategic character. Now, Boolean games are computationally complex: checking whether a Boolean game has a pure strategy Nash equilibrium is Σ_2^p-complete, while checking whether a particular outcome is a pure Nash equilibrium is co-NP-complete. If Boolean games are to find applications, then the issue of intractability must surely be addressed.

Our contribution in the present paper is twofold. Following the discussion above, we first formally define and investigate the notion of *k-bounded Nash equilibrium* for Boolean games. An outcome is a k-bounded Nash equilibrium if no agent has any incentive to deviate by flipping the value of at most k variables. We consider the computational aspects of k-bounded equilibria, and then prove a logical characterisation of those classes of Boolean games for which k-bounded Nash equilibria and standard pure strategy Nash equilibria coincide. We then move on to consider classes of Boolean games for which the computation of pure strategy Nash equilibria is easier than the general case. Again, we give a logical characterisation of such cases. We conclude with discussion and some issues for future research.

2. GAMES AND SOLUTION CONCEPTS

Let $\mathcal{G}$ be a class of games, and let Ω be the set of *outcomes* for these games; for the purposes of this discussion, it does not matter exactly what the games and outcomes are. A *solution concept* for $\mathcal{G}$ can be understood as a function:

$$\sigma : \mathcal{G} \to \mathbf{2}^{\Omega}.$$

That is, a solution concept identifies with every game a subset of outcomes; intuitively, those that are "rational" according to the solution concept in question.

The obvious computational problems associated with such a solution concept σ for a class of games $\mathcal{G}$ are as follows:

- NON-EMPTINESS:
  Given some $G \in \mathcal{G}$, is it the case that $\sigma(G) \neq \emptyset$?
- MEMBERSHIP:
  Given $G \in \mathcal{G}$ and $\omega \in \Omega$, is it the case that $\omega \in \sigma(G)$?
- COMPUTATION:
  Given $G \in \mathcal{G}$, exhibit some ω such that $\omega \in \sigma(G)$.

In the present paper, we will be concerned largely with the first two of these problems. Now, for many important classes of games, these problems are computationally hard [3, 12, 4, 15, 8]. As mentioned in the introduction, there are at least two ways to approach this problem:

1. *Develop alternative solution concepts, which lend themselves to being computed efficiently.* For example, the notion of ϵ-Nash equilibrium has been developed, which relaxes the strict notion of Nash equilibrium by requiring that no player can gain more than ϵ in a deviation from a strategy profile [13, p.83].
2. *Try to identify useful classes of games or representations of games for which solution concepts can be easily computed.* In cooperative game theory, for example, the marginal contribution net representation allows for the efficient computation of the Shapley value solution concept [10].

With respect to the first proposal, let us make the discussion a little more formal. Suppose we have a solution concept σ for a class of games $\mathcal{G}$ such that the associated computational problems (NON-EMPTINESS, MEMBERSHIP, COMPUTATION) are intractable (NP-hard or worse). Then we might try to develop an alternative solution concept $\hat{\sigma}$, which "approximates" σ, but which is tractable. Note that here we mean "approximate" in the informal everyday sense, rather than the formal sense of approximation algorithms and FPTAS [1] (although of course looking for FPTAS would be a very natural approach). Now, how might σ and its "approximation", $\hat{\sigma}$, be related? We can consider two natural properties, as follows.

- We say $\hat{\sigma}$ is a *sound approximation* of a solution concept σ for a class of games $\mathcal{C} \subseteq \mathcal{G}$ if
  $$\forall G \in \mathcal{C} : \quad \hat{\sigma}(G) \subseteq \sigma(G)$$
  i.e., the solutions proposed by $\hat{\sigma}$ are a subset of those proposed by σ.
- We say $\hat{\sigma}$ is a *complete approximation* of a solution concept σ for a class of games $\mathcal{C} \subseteq \mathcal{G}$ if
  $$\forall G \in \mathcal{C} : \quad \hat{\sigma}(G) \supseteq \sigma(G)$$
  i.e., the solutions proposed by $\hat{\sigma}$ are a superset of those proposed by σ.

In the limit, where $\hat{\sigma}$ is a sound and complete approximation of σ w.r.t. the class $\mathcal{C} = \mathcal{G}$ of all games, then σ and $\hat{\sigma}$ would be identical, and it would then be no easier to compute $\hat{\sigma}$ than σ. A typical situation, with respect to the class $\mathcal{G}$ of all games, is that we will have approximate solution concepts $\hat{\sigma}$ that are complete (all solutions according to σ are solutions according to $\hat{\sigma}$) but not sound (not all solutions according to $\hat{\sigma}$ are solutions according to σ). This is exactly the situation with ϵ-Nash equilibrium, for example: all "exact" Nash equilibria are ϵ-Nash equilibria, but in general, (i.e., where $\epsilon > 0$), not all ϵ-Nash equilibria will be "exact" Nash equilibria. The fact that an approximate solution concept is complete but not sound captures our intuitions about relaxing the requirements for optimality inherent in exact solution concepts: an approximate solution concept will often admit more solutions than its exact counterpart, thus (we hope) making approximate solutions easier to find.

If we have an approximate solution concept $\hat{\sigma}$, then one interesting and important question is the following: can we identify a class of games $\mathcal{C} \subset \mathcal{G}$ such that $\hat{\sigma}$ is a sound and complete approximation to σ with respect to $\mathcal{C}$, *even though $\hat{\sigma}$ is not sound and complete with respect to the class of all games $\mathcal{G}$*? If we can do this, and the class $\mathcal{C}$ corresponds to games that are of practical value, then this means that the approximate solution concept is in fact all we need: we do not need to look for exact solutions σ, since these will in any case be given by $\hat{\sigma}$.

In the present paper, we will focus on *bounded* approximations to Nash equilibria for Boolean games, which will be complete but not sound with respect to the class of all games, and we will identify classes of games for which the bounded solution concept is both sound and complete.

3. BOOLEAN GAMES

We now present the formal framework of propositional logic and Boolean games that we use throughout the remainder of this paper. Our presentation is fairly standard [9, 2, 6, 7].

Propositional Logic: Let $\mathbb{B} = \{\top, \bot\}$ be the set of Boolean truth values, with "$\top$" being truth and "$\bot$" being falsity. We will abuse notation a little by using $\top$ and $\bot$ to denote both the syntactic constants for truth and falsity respectively, as well as their semantic counterparts. Let $\Phi = \{p, q, \ldots\}$ be a (finite, fixed, non-empty) vocabulary of Boolean variables, and let $\mathcal{L}$ denote the set of (well-formed) formulae of propositional logic over Φ, constructed using the conventional Boolean operators ("$\wedge$", "$\vee$", "$\rightarrow$", "$\leftrightarrow$", and "$\neg$"), as well as the truth constants "$\top$" and "$\bot$". Where $\varphi \in \mathcal{L}$, we let $\mathit{vars}(\varphi)$ denote the (possibly empty) set of Boolean variables occurring in φ (e.g., $\mathit{vars}(p \wedge q) = \{p, q\}$). A *valuation* is a total function $v : \Phi \rightarrow \mathbb{B}$, assigning truth or falsity to every Boolean variable. We write $v \models \varphi$ to mean that the propositional formula φ is true under, or satisfied by, valuation v, where the satisfaction relation "$\models$" is defined in the standard way. Let $\mathcal{V}$ denote the set of all valuations over Φ. We write $\models \varphi$ to mean that φ is a tautology. We denote the fact that $\models \varphi \leftrightarrow \psi$ by $\varphi \equiv \psi$. We use some additional definitions, as follows:

- A *literal*, ℓ, is either (i) a Boolean variable or the negation of a Boolean variable, or (ii) a Boolean constant (i.e., a member of $\mathbb{B}$) or the negation of a Boolean constant.
- A *clause*, C, is a disjunction of literals, i.e., a formula of the form $C = \ell_1 \vee \cdots \vee \ell_m$.
- A *Horn clause* is a clause in which at most one literal is not negated.
- A formula is in *Conjunctive Normal Form* (CNF) if it is a conjunction of clauses, i.e., is of the form $\varphi = C_1 \wedge \cdots \wedge C_l$, where each C_i, $(1 \le i \le l)$ is a clause.
- A CNF formula $\varphi = C_1 \wedge \cdots \wedge C_l$ is in u-CNF, if each clause C_i $(1 \le i \le l)$ contains at most u literals.
- A CNF formula $\varphi = C_1 \wedge \cdots \wedge C_l$ is in u-clause CNF if it contains no more than u clauses (i.e., $l \le u$).

  Notice that there is an important difference between u-CNF and u-clause CNF: the former constrains the number of literals permitted in a clause, but does not constrain the number of clauses permitted in a formula; while the latter constrains the number of clauses permitted in a formula, but does not constrain the number of literals that appear in clauses.
- A CNF formula $\varphi = C_1 \wedge \cdots \wedge C_l$ is said to be in Horn clause form if for all $1 \le i \le l$, the clause C_i is a Horn clause.

The *satisfiability problem* for formulae φ is the problem of determining whether there exists a valuation v such that $v \models \varphi$. For arbitrary CNF formulae, this problem is of course NP-complete; for 2-CNF formulae, and for Horn clause formulae, the satisfiability problem is decidable in polynomial time (see, e.g., [11]).

Agents and Variables: The games we consider are populated by a set $N = \{1, \ldots, n\}$ of *agents* – the players of the game. Each agent is assumed to have a *goal*, characterised by an $\mathcal{L}$-formula: we write γ_i to denote the goal of agent $i \in N$. Agents $i \in N$ each *control* a (possibly empty) subset Φ_i of the overall set of Boolean variables. By "control", we mean that i has the unique ability within the game to set the value (either $\top$ or $\bot$) of each variable $p \in \Phi_i$. We will require that $\Phi_i \cap \Phi_j = \emptyset$ for $i \neq j$, and that $\Phi_1 \cup \cdots \cup \Phi_n = \Phi$ (i.e., $\Phi_1, \ldots, \Phi_n$ partition Φ).

When playing a Boolean game, the primary aim of an agent i will be to choose an assignment of values for the variables Φ_i under its control so as to satisfy its goal γ_i. The difficulty is that γ_i may contain variables controlled by other agents $j \neq i$, who will also be trying to choose values for their variables Φ_j so as to get their goals satisfied; and their goals in turn may be dependent on the variables Φ_i. A *choice* for agent $i \in N$ is a function $v_i : \Phi_i \rightarrow \mathbb{B}$, i.e., an allocation of truth or falsity to all the variables under i's control. Let $\mathcal{V}_i$ denote the set of choices for agent i.

Outcomes: An *outcome* is a collection of choices, one for each agent. Formally, an outcome is a tuple $(v_1, \ldots, v_n) \in \mathcal{V}_1 \times \cdots \times \mathcal{V}_n$. Notice that an outcome defines a value for all variables, and we will often think of outcomes as valuations, for example writing $(v_1, \ldots, v_n) \models \varphi$ to mean that the valuation defined by the outcome $(v_1, \ldots, v_n)$ satisfies formula $\varphi \in \mathcal{L}$.

Boolean Games: A Boolean game, G, is a $(2n+2)$-tuple:

$$G = \langle N, \Phi, \Phi_1, \ldots, \Phi_n, \gamma_1, \ldots, \gamma_n \rangle$$

where $N = \{1, \ldots, n\}$ is a set of agents, $\Phi = \{p, q, \ldots\}$ is a finite set of Boolean variables, $\Phi_i \subseteq \Phi$ is the set of Boolean variables under the unique control of $i \in N$, and $\gamma_i \in \mathcal{L}$ is the goal of agent $i \in N$.

Utility: We now introduce a model of utility for our games. The basic idea is that an agent will strictly prefer all outcomes in which it gets its goal achieved over all outcomes where it does not. We capture this in utility functions $u_i(\cdots)$ defined over outcomes $(v_1, \ldots, v_n)$:

$$u_i(v_1, \ldots, v_n) = \begin{cases} 1 & \text{if } (v_1, \ldots, v_n) \models \gamma_i \\ 0 & \text{otherwise.} \end{cases}$$

Nash Equilibrium: Let $(v_1, \ldots, v_i, \ldots, v_n)$ be an outcome. We say that a player i has a *beneficial deviation* if there exists a choice $v_i' \in \mathcal{V}_i$ for i such that $u_i(v_1, \ldots, v_i', \ldots, v_n) > u_i(v_1, \ldots, v_i, \ldots, v_n)$. In this case, v_i' serves as a witness to the beneficial deviation. It will be useful to refer to the following fact later.

OBSERVATION 1. *Suppose $(v_1, \ldots, v_i, \ldots, v_n) \in \mathcal{V}_1 \times \cdots \times \mathcal{V}_i \times \cdots \times \mathcal{V}_n$ and $v_i' \in \mathcal{V}_i$. Then v_i' is a beneficial deviation for i from $(v_1, \ldots, v_i, \ldots, v_n)$ iff:*

1. $(v_1, \ldots, v_i, \ldots, v_n) \not\models \gamma_i$ *and*
2. $(v_1, \ldots, v_i', \ldots, v_n) \models \gamma_i$.

We then say an outcome $(v_1, \ldots, v_n)$ is a Nash equilibrium if no player has a beneficial deviation. We denote the Nash equilibrium outcomes of a game G by $NE(G)$. It may of course be that $NE(G) = \emptyset$.

Referring back to our discussion in the preceding section, there are two obvious decision problems relating to Nash equilibria in Boolean games: NON-EMPTINESS (given a game G, is it the case that $NE(G) \neq \emptyset$?) and MEMBERSHIP (given a game G and an outcome $(v_1, \dots, v_n)$ for G, is it the case that $(v_1, \dots, v_n) \in NE(G)$?). Unfortunately, it is known that both of these problems are computationally complex [2]:

PROPOSITION 1. *The* NON-EMPTINESS *problem for Boolean games is Σ_2^p-complete; the* MEMBERSHIP *problem for Boolean games is co-NP-complete.*

4. K-BOUNDED EQUILIBRIA

We will now define a new class of equilibria for Boolean games, which we will call *k-bounded equilibria*. To motivate this new class, take an agent i who is trying to decide whether he has any beneficial deviation from an outcome $(v_1, \dots, v_i, \dots, v_n)$. From Proposition 1, this problem is in general NP-complete. Intuitively, player i is trying to decide whether he can get his goal achieved by flipping the value of some subset of his variables. The complexity in this problem arises because i must consider $2^{|\Phi_i|}$ sets of variables in this evaluation. It follows that our agent's task will be simpler if we can eliminate some of these sets of variables from player i's consideration.

Where $\{v_i, v_i'\} \subseteq \mathcal{V}_i$ are choices for player $i \in N$, let us define the *distance* between them as being the number of variables that have different values in the two valuations; we denote this value by $\delta(v_i, v_i')$:

$$\delta(v_i, v_i') = |\{x \in \Phi_i \mid v_i(x) \neq v_i'(x)\}|.$$

We sometimes refer to $\delta(v_i, v_i')$ as the *size* of the deviation v_i'. Now, given an outcome $(v_1, \dots, v_i, \dots, v_n)$ and a value $k \in \mathbb{N}, k > 1$ we will say a player i has a *k-bounded beneficial deviation* if there is some $v_i' \in \mathcal{V}_i$ such that:

1. $\delta(v_i, v_i') \leq k$; and
2. $u_i(v_1, \dots, v_i', \dots, v_n) > u_i(v_1, \dots, v_i, \dots, v_n)$.

We will say an outcome $(v_1, \dots, v_i', \dots, v_n)$ is a *k-bounded Nash equilibrium* if no player has a k-bounded beneficial deviation. Let the set of k-bounded Nash equilibria of game G be denoted by $NE_k(G)$.

EXAMPLE 1. *Consider the Boolean game,*

$$G^{(t)} = \langle \{a_1, a_2\}, \Phi, \Phi_1, \Phi_2, \gamma_1, \gamma_2 \rangle$$

in which $\Phi_1 = \{x_1, \dots, x_t\}$, $\Phi_2 = \{y_1, \dots, y_t\}$ $(t \geq 1)$ and

$$\gamma_1 = \left(\bigvee_{i=1}^{t} y_i \right) \wedge \left(\bigwedge_{j=1}^{t} x_j \right)$$

$$\gamma_2 = \left(\bigvee_{i=1}^{t} x_i \right) \wedge \left(\bigwedge_{j=1}^{t} y_j \right)$$

The outcome v in which $x_1 = y_1 = \top$, and $x_i = \bot$, $y_i = \bot$ for all $i \neq 1$, is a k-bounded equilibrium for all $k < t - 1$. It is not, however, a Nash equilibrium: neither goal is satisfied but by changing the $t - 1$ variables under its control to $\top$ both agents can realise their goals.

How does the notion of k-bounded equilibrium relate to the general concept of Nash equilibrium? We have:

PROPOSITION 2. *The solution concept of k-bounded Nash equilibrium is a complete but unsound approximation for pure strategy Nash equilibria in Boolean games. Formally:*

1. *There exist Boolean games G and bounds $k \in \mathbb{N}, k \geq 1$ such that $NE_k(G) \not\subseteq NE(G)$.*
2. *For all Boolean games G and bounds $k \in \mathbb{N}, k \geq 1$, we have $NE(G) \subseteq NE_k(G)$.*

PROOF. Example 1 illustrates point (1) (in fact, it is easily seen that the construction of Example 1 shows that for all $k \geq 2$ there are games in which $NE_{k-1}(G) \subset NE_k(G)$, i.e. k-bounded equilibria are more general than $(k-1)$-bounded). For point (2), observe that if an outcome is stable in the general sense, then no player has any beneficial deviation; and in particular, no player has any beneficial deviation of size $\leq k$. □

Now, for the notion of k-bounded equilibrium to be of anything other than purely theoretical interest, it must reduce the complexity of the reasoning task faced by agents. Of course, it is easy to see that, with respect to worst case asymptotic analysis, k-bounded Nash equilibria present no advantages over Nash equilibria, since if we set the bound k to

$$k = \max\{|\Phi_i| \mid i \in N\}$$

then k-bounded Nash equilibrium collapses to the standard notion of Nash equilibrium for Boolean games. However, we will now show that nevertheless, by constraining possible deviations v_i' such that $\delta(v_i, v_i') \leq k$, we can dramatically reduce the search space of possible deviations. Formally, where G is a game containing a player i, let $v_i \in \mathcal{V}_i$ be a choice for i, and let $k \in \mathbb{N}, k \geq 1$ be a bound, then we have:

$$|\{v_i' \in \mathcal{V}_i \mid \delta(v_i, v_i') \leq k\}| = \sum_{j=1}^{k} \binom{|\Phi_i|}{j}.$$

From standard combinatorics, it follows that if we set the deviation bound k so that $k < |\Phi_i|/2$ (for example), then the search space will for a beneficial deviation will be less than half the search space for general deviations. We thus have an exponential reduction in the size of the search space when looking for k-bounded beneficial deviations, compared to the case for pure Nash equilibria in general.

In fact, from recent results of Szeider [14] it turns out that this upper bound can often be signifcantly improved.

PROPOSITION 3. *Let G be a Boolean game in which every goal formula, γ_i, is expressed as a CNF formula containing at most t clauses, where t is a arbitrary but fixed natural number, and v be an outcome. Deciding if v is a k-bounded equilibrium is fixed-parameter tractable (with respect to the parameter k), i.e., there is a decision algorithm whose running time is bounded above by $f(k)$ $poly(|G|)$, where $|G|$ is the number of bits needs to encode the game G, and $f(\cdots)$ is a function whose value depends* only *on k.*

PROOF. Szeider [14] shows that instances of the so-called "k-FLIP SAT" problem, whereby given a CNF formula, F, and assignment, α to its variables, it is required to decide if there is some assignment β satisfying F and having $\delta(\beta, \alpha) \leq k$, may be decided in $(f(k) + t^3)poly(|F|)$ steps. The "k-FLIP SAT" problem, however, is exactly that of deciding if γ_i, after simplification to $\gamma_i^{\Phi_i}$, i.e., the CNF formula arising by applying the variable settings to $\Phi \setminus \Phi_i$, is satisfiable by changing the values of (exactly) k variables. We can thus decide if i has a k-bounded beneficial deviation just by considering each $1 \leq l \leq k$, deciding if $\gamma_i^{\Phi_i}$ is a positive instance of l-FLIP SAT with respect to the assignment v_i of values currently set in Φ_i. In total, v will be accepted as a k-bounded equilibrium if no agent suceeds in identifying a beneficial deviation via this process. $\square$

Now, it might seem that k-bounded equilibria are much weaker than general Nash equilibria, in the sense that a player may well have a beneficial deviation from an outcome in the general case, but not if we restrict ourselves to k-bounded Nash equilibria. But in fact, in some classes of games, this is not an issue: the following proposition shows that, in certain useful and important classes of games, *general Nash equilibria and k-bounded Nash equilibria coincide.*

PROPOSITION 4. *The solution concept of k-bounded Nash equilibrium is a sound and complete approximation of pure strategy Nash equilibria for the class of Boolean games in which every agent has a goal formula that is logically equivalent to a propositional formula in k-clause CNF. In other words, for all Boolean games*

$$G = \langle N, \Phi, \Phi_1, \ldots, \Phi_n, \gamma_1, \ldots, \gamma_n \rangle,$$

if there exists some $k \in \mathbb{N}, k \geq 1$ such that for every agent $i \in N$ there exists a k-clause CNF formula γ_i' such that $\gamma_i \equiv \gamma_i'$, then we have $NE(G) = NE_k(G)$.

PROOF. Consider an arbitrary outcome $(v_1, \ldots, v_i, \ldots, v_n)$ and a player $i \in N$. We claim that player i has a k-bounded beneficial deviation iff the player has a general beneficial deviation. The left-to-right implication is obvious. For the right-to-left implication, suppose the player has a beneficial deviation, call it v_i'. Then from Observation 1, we know that:

- $(v_1, \ldots, v_i, \ldots, v_n) \not\models \gamma_i$
- $(v_1, \ldots, v_i', \ldots, v_n) \models \gamma_i$

From the conditions of the Proposition, we can infer:

- $(v_1, \ldots, v_i, \ldots, v_n) \not\models \gamma_i'$
- $(v_1, \ldots, v_i', \ldots, v_n) \models \gamma_i'$.

The existence of a k-bounded beneficial deviation may then be seen as follows. Since γ_i' is in k-clause CNF, it is the conjunction of no more than k clauses: $\gamma_i' = C_1 \wedge \cdots \wedge C_k$, where each C_i is a disjunction of literals. The assignment v_i' need only satisfy at most one literal from each clause C_i, and so to satisfy γ_i', (and hence γ_i), we only need to flip at most k variables compared to v_i. $\square$

Now, the implication of this result is that if agents have goals that are logically equivalent to k-clause CNF formulae, then *we don't need to consider arbitrary deviations: k-bounded deviations are all we need.* It follows that in such games, the search space for possible deviations can be dramatically reduced, compared to general Nash equilibria.

As an aside, observe that *every* propositional logic formula can be converted into CNF. So, if we start with a Boolean game in which goals are arbitrary propositional formulae, we can translate each formula into an equivalent CNF form, and take k to be the largest number of clauses of any CNF formula in the resulting game. If we end up with $k < \max\{|\Phi_i| \mid i \in N\}$, then this tells us that we can rule out a potentially large fraction of the search space when looking for beneficial deviations, as we only have to consider k-bounded Nash equilibria. However, in the worst case, translation to CNF can result in an exponential blow-up in the number of clauses in the formula: $k = O(2^{|\Phi|})$. In such cases, we would clearly obtain no benefit from k-bounded equilibria.

5. TRACTABLE BOOLEAN GAMES

An alternative to developing solution concepts that are easy (or at least, easier) to compute than exact solutions is to consider classes of games for which exact solution concepts are easy to compute. In our case, consider the problem of determining whether a player $i \in N$ has a beneficial deviation in a Boolean game. This involves the player considering $2^{|\Phi_i|}$ choices, to see whether it can get its goal achieved through making one of these. So, to what extent can we identify classes of games for which checking for beneficial deviations is computationally easy? Well, as a starting point, the following is very easy to see:

OBSERVATION 2. *Let $\mathcal{C} \subseteq \mathcal{L}$ be a class of propositional logic formulae with a polynomial time satisfiability problem. The* MEMBERSHIP *problem for $G = \langle N, \Phi, \Phi_1, \ldots, \Phi_n \gamma_1, \ldots, \gamma_n \rangle$ with $\{\gamma_1, \ldots, \gamma_n\} \subseteq \mathcal{C}$ is decidable in polynomial time, and the* NON-EMPTINESS *problem is in NP.*

So, for example, if the goal formulae of all players are expressed in, e.g., Horn clause form, or 2-CNF, then checking whether an outcome is a Nash equilibrium or not is polynomial time decidable, and checking whether there exists a stable outcome is in NP. However, as we will now see, we can in fact significantly strengthen this result; to do this, however, we need some further notation and terminology.

First, where $S = \{1, \ldots, k\}$ is a subset of players and $\{v_1, \ldots, v_k\}$ is a collection of choices, one for each player $i \in S$, define a function w by:

$$w_{\{v_1,\ldots,v_k\}}(x) = \begin{cases} v_1(x) & \text{if } x \in \text{dom } v_1 \\ v_2(x) & \text{if } x \in \text{dom } v_2 \\ \cdots & \cdots \\ v_k(x) & \text{if } x \in \text{dom } v_k \end{cases}$$

Where φ is a propositional logic formula, and $\{v_1, \ldots, v_k\}$ is a collection of choices, one for each player in $S = \{1, \ldots, k\}$, then we will denote by $\varphi[v_1, \ldots, v_k]$ the formula obtained from φ by systematically replacing each variable x such that

$$x \in \text{dom } v_1 \cup \cdots \cup \text{dom } v_k$$

by the Boolean value $w_{\{v_1,\ldots,v_k\}}(x)$. Finally, the *reduction* of $\varphi[v_1, \ldots, v_k]$ will be denoted by $\varphi^*[v_1, \ldots, v_k]$, and is defined to be the propositional logic formula obtained from $\varphi[v_1, \ldots, v_k]$ by carrying out the following two steps:

1. deleting any clause containing $\top$ or $\neg\bot$;
2. deleting $\bot$ or $\neg\top$ from any clause in which they occur.

If the resulting formula contains any empty clause (i.e., all literals have been deleted from the original clause), then we replace the whole formula by $\bot$.

The soundness of these simplification steps is clear from basic propositional reasoning.

Now, given a player i's goal γ_i, we will denote the *range* of γ_i by $rng(\gamma_i)$, and define this to be the following set of formulae:

$$rng(\gamma_i) = \{\gamma_i^*[v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n] \mid (v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n) \in \prod_{i \in N \setminus \{i\}} \mathcal{V}_i\}.$$

Thus, intuitively, $rng(\gamma_i)$ is the set of formulae that could be obtained from γ_i by simplifying it under all possible combinations of choices made by other players. Notice that if $\psi \in rng(\gamma_i)$, then the only variables occurring in ψ will be controlled by i, i.e., for each $\psi \in rng(\gamma_i)$, we have $vars(\psi) \subseteq \Phi_i$.

Now, let $\mathcal{C} \subseteq \mathcal{L}$ be a class of propositional logic formulae, and let $G = \langle N, \Phi, \Phi_1, \ldots, \Phi_n, \gamma_1, \ldots, \gamma_n \rangle$ be a Boolean game. Then we say *the range of G is in $\mathcal{C}$* if:

$$rng(\gamma_i) \subseteq \mathcal{C} \quad \text{for all } i \in N.$$

Given this, we can strengthen Observation 2 as follows.

PROPOSITION 5. *Let $\mathcal{C} \subseteq \mathcal{L}$ be a class of propositional logic formulae with a polynomial time satisfiability problem, and let $G = \langle N, \Phi, \Phi_1, \ldots, \Phi_n, \gamma_1, \ldots, \gamma_n \rangle$ be a game such that the range of G is in $\mathcal{C}$. Then the* MEMBERSHIP *problem for G is decidable in polynomial time and the* NON-EMPTINESS *problem is in NP.*

Notice Proposition 5 is *not* the same as Observation 2, above (although it is related). It is a much stronger result: it *does not* require that the goal formulae γ_i are in a tractable form; only that *the range of the goal formulae are tractable*, (i.e., the formulae obtained by simplifying the goal formulae under the possible choices for all other players). This is a very different, and much more powerful result than that of Observation 2. In particular, for every player $i \in N$, the only constraints it imposes on the goal formulae of player i relate to the variables actually controlled by player i; essentially no constraints are placed on the variables of players $j \neq i$. Thus we obtain:

PROPOSITION 6. *For the following classes of games, the* MEMBERSHIP *problem is decidable in polynomial time, while the* NON-EMPTINESS *problem is NP-complete:*

1. *Games in which for all players $i \in N$, if $\gamma_i = C_1 \wedge \cdots \wedge C_l$, then we have $|vars(C_j) \cap \Phi_i| \leq 2$ for all $1 \leq j \leq l$.*
2. *Games in which for all players $i \in N$, if $\gamma_i = C_1 \wedge \cdots \wedge C_l$, then for all $1 \leq j \leq l$, the clause C_j contains at most one unnegated element of Φ_i.*

PROOF. We will do the proof for point (1); the second point is similar. So consider MEMBERSHIP. Take an arbitrary outcome $(v_1, \ldots, v_{i-1}, v_i, v_{i+1}, \ldots, v_n)$.

Since for all players $i \in N$, if $\gamma_i = C_1 \wedge \cdots \wedge C_l$, then we have $|vars(C_j) \cap \Phi_i| \leq 2$ for all $1 \leq j \leq l$, then it follows that $\gamma_i^*[v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n]$ is in 2-CNF, and contains only variables in Φ_i. From Proposition 1 we can see that i has a beneficial deviation from

$$(v_1, \ldots, v_{i-1}, v_i, v_{i+1}, \ldots, v_n)$$

if there exists a choice $v_i' \in \mathcal{V}_i$ such that

$$v_i' \models \gamma_i^*[v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n].$$

Since $\gamma_i^*[v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n]$ is in 2-CNF, this check can be done in polynomial time. We now prove NP-completeness of the NON-EMPTINESS problem. Membership is obvious by "guess and check". For hardness, we reduce SAT. Let φ be a SAT instance. We construct a game G_φ, satisfying the conditions of the proposition, such that $NE(G_\varphi) \neq \emptyset$ iff φ is satisfiable. Assume φ has l clauses, $\varphi = C_1 \wedge \cdots \wedge C_l$, and $vars(\varphi) = \{x_1, \ldots, x_k\}$. We introduce one additional variable, z. We define $k+1$ agents, with:

- player $1 \leq i \leq k$ controlling variable x_i and having goal y; and
- player $k+1$ controlling variable y and having goal
$$\gamma_{k+1} = (\varphi \wedge z) \vee \neg(y \leftrightarrow z).$$
- player $k+2$ controls variable z and has goal
$$\gamma_{k+2} = (\neg\varphi) \wedge (y \leftrightarrow z)$$

(Notice that γ_{k+1} and γ_{k+2} can trivially be translated into the form required by the Proposition.) We claim that φ is satisfiable iff $N(G_\varphi) \neq \emptyset$. If φ is satisfiable, then take any satisfying assignment for φ and set $y = z = \top$. We claim that this outcome is stable: for observe that in this case, players $1 \leq i \leq k+1$ have their goal achieved, and so cannot benefit by deviating; and player $k+2$ does not get his goal achieved, but has no beneficial deviation. Now we claim that if φ is unsatisfiable then $NE(G_\varphi) = \emptyset$. For consider any outcome. First observe that since φ is unsatisfiable, any outcome will falsify φ. Thus, player $k+2$ will have his goal achieved if the outcome assigns y and z the same value; if the outcome does not give them the same value then player $k+2$ has a beneficial deviation. However, if player $k+2$ deviates to give y and z the same value, then player $k+1$ would have a beneficial deviation. Hence no outcome can be a Nash equilibrium. □

5.1 Equilibria that Maximise Social Welfare

The results above indicate that, in the event that $rng(\gamma_i)$ falls within a class of formulae that have a polynomial time satisfiability problem, then determining whether a given outcome is an equilibrium for the instance is tractable. It may, however, often be the case that we do not merely wish to accept *any* equilibrium, but would, if possible, prefer to identify one which satisfies some notion of "optimality". In the case of Boolean games a natural way of distinguishing between equilibria is to associate each with its *social welfare*, which we define as the total number of agents whose goal is satisfied. As a very simple example of a Boolean game in which there are equilibria in which this measure varies significantly, consider $n = 2m$ agents, $N = \langle a_1, \ldots, a_m, b_1, \ldots, b_m \rangle$ each of which has control over exactly one propositional variable from

$$\Phi = \langle x_1, \ldots, x_m, y_1, \ldots, y_m \rangle$$

so that i determines the value of x_i and b_i that of y_i. Suppose the goal formula for i is $x_i \wedge y_i$, and that for b_i is $x_i \vee \neg y_i$. It is easily checked that the outcome in which all variables are assigned $\perp$ is a Nash equilibrium for this game that satisfies *only* the goals of the b_i agents. On the other hand, the assignment in which all variables are assigned $\top$ is also a Nash equilibrium, but one which that satisfies the goals of *all* agents. Intuitively, the latter seems preferable to the former as a solution.[1] This motivates the following decision question, which we present, at first, in its most general form:

> GOAL MAXIMIZATION (GM):
> Given a Boolean game, G involving n agents and $t \in \mathbb{N}$ such that $1 \leq t \leq n$, is there an outcome v for which $v \in \mathcal{N}(G)$ *and* $|\{\gamma_i \ : \ v \models \gamma_i\}| \geq t$?

It is not hard to show that, even if G admits a polynomial time process for deciding membership, this is not sufficient to ensure GM is tractable.

PROPOSITION 7. *GM is NP-complete even if instances are restricted to those in which every goal formula is in 2-CNF, which we will denote* GM^2.

PROOF. That GM is in NP follows by simply guessing an outcome, v, and checking that v satisfies at least t goals. For NP-hardness, we recall that the so-called MAX-2-SAT problem – deciding if a given 2-CNF, F, has an assignment that satisfies at least K of its clauses – is NP–complete. We show that MAX-2-SAT is polynomially reducible to GM^2.

Given $\langle F, K\rangle$ an instance of MAX-2-SAT in which F uses n variables, $\{x_1, \ldots, x_n\}$ and has r clauses, form an instance, $\langle G_{\langle F,K\rangle}, t\rangle$ of GM^2 as follows: $G_{\langle F,K\rangle}$ has $r+1$ agents, with $\gamma_i \ = \ C_i$ the i'th clause of F for $1 \leq i \leq r$, and $\gamma_{r+1} \equiv \top$. Set $\Phi = \{x_1, \ldots, x_n\}$, $\Phi_i = \emptyset$ when $1 \leq i \leq r$ and $\Phi_{r+1} = \Phi$. To complete the instance t is fixed to $K+1$. We claim that $\langle G_{\langle F,K\rangle}, t\rangle$ is a positive instance of GM^2 iff $\langle F, K\rangle$ is a positive instance of MAX-2-SAT. Trivially, any assigment α to $\langle x_1, \ldots, x_n\rangle$ that satisfies at least K clauses of F, immediately yields an outcome that achieves the goals of the corresponding K agents and since γ_{r+1} is always satisfied this outcome satisfies $t = K+1$ goals. Furthermore the outcome is in $\mathcal{N}(G_{\langle F,K\rangle})$: i $(1 \leq i \leq r)$ cannot deviate (no variables are under its control) and a_{r+1} has no reason to deviate (its goal is already satisfied). Conversely, should there be an outcome $v \in \mathcal{N}(G_{\langle F,K\rangle})$ satisfying at least $t = K+1$ goals then, since this outcome must satisfy γ_{r+1} it follows that at least K goals from $\{\gamma_1, \ldots, \gamma_r\}$ are satisfied so that the corresponding assignment witnesses $\langle F, K\rangle$ as a positive instance of MAX-2-SAT. $\square$

5.2 Utility as Reachability

The forms taken by utility functions as described are somewhat restrictive in that these fail to model the possibility that (assuming, say, negotiation with other agents can take place) despite not having its goal satisfied with a particular outcome, an agent may, in fact be "close to" realising its intended goal. To make this idea more precise, suppose we define the *t-reachable utility* of an outcome to i (denoted w_i^t to distinguish from our standard notion of utility u_i) by

$$w_i^t(v) \ = \ \begin{cases} 1 \ - \ \min\{ \ r \ : \ r \leq t \ \text{ and } \exists v' \text{ with } \delta(v, v') = r \text{ and } v' \models \gamma_i\}/|\Phi| \\ 0 \quad \text{if no suitable } v' \text{ exists} \end{cases}$$

Notice that for any outcome, v, $w_i^t(v) \geq w_i^0(v) = u_i(v)$ and captures the behaviour that there is a *possibility* of γ_i being achieved (even though this may not be completely within i's control).

It turns out, although this new form appears superficially computationally more demanding, if we limit attention to k-bounded deviations then we can still identify tractable versions of the membership problem.

[1]Endriss *et al.* discuss taxation-based mechanisms that incentivise players to socially desirable Nash equilibria [7].

PROPOSITION 8. *Let $\mathcal{C}$ be any class of propositional formulae for which k-FLIP SAT is fixed-parameter tractable wrt to parameter k. Let $t \in \mathbb{N}$ be fixed, G be a Boolean game with agent utilities captured through w_i^t. If all goal formulae are in the class $\mathcal{C}$ then for any outcome, $v \in NE_t(G)$ is polynomial time decidable.*

PROOF. Suppose that v is an outcome. By definition $v \in NE_t(G)$ iff no i has a t-bounded beneficial deviation, i.e. letting $v//\alpha$ denote the outcome obtained from v by the values currently assigned to Φ_i in v being replaced by α, it follows that $v \in NE_t(G)$ if and only if for each i:

$$\forall \ \alpha \in \langle \top, \bot\rangle^{|\Phi_i|} \ \delta(v, v//\alpha) \leq t \ \Rightarrow \ w_i^t(v//\alpha) \leq w_i^t(v)$$

In order to test if i has a beneficial deviation (under the new notion of utility) it is first necessary to compute $w_i^t(v)$, i.e. to determine if by changing the values of $0, \ 1, \ldots, t$ variables in the outcome v it is the case that γ_i can be achieved. This, however, is simply the r-FLIP SAT problem and the goal formulae are restricted to those within some fixed-parameter tractable (with parameter r) class. Thus we can compute $w_i^t(v)$ efficiently and it remains only to test if there is a t-bounded deviation which improves upon this, i.e whether by altering at most t variables within the control of i (leaving the values assigned to other variables unchanged), it is possible to to construct an outcome v' for which $w_i^t(v') > w_i^t(v)$. this, however, can (at worst) be carried out just by enumerating through the $O(|\Phi_i|^t)$ possible deviations. $\square$

We note, however, that very simple negotiation protocols, even when only two agents are involved may lead to problematic situations. For example consider the following. We have a Boolean game, $G = \langle \{a_1, a_2\}, \Phi, \Phi_1, \Phi_2, \gamma\rangle$ in which the following protocol, which call the *agreed-1-flip* protocol is used: starting from initial assignments $\underline{a}$ (for Φ_1) and $\underline{b}$ (for Φ_2) a_1 proposes a variable of Φ_2 to flip at the same time as a_2 proposes a variable of Φ_1 for a_1 to flip. If γ (the common goal for both agents) is not satisfied the process continues. Although this mechnaism is very basic, it turns out – even for γ being simply a conjunction of literals, that starting from an ill-chosen initial assignment can lead to exponentially many negotiation rounds taking place. That is,

PROPOSITION 9. *There are 2-player Boolean games, $G = \langle \{a_1, a_2\}, \Phi, \Phi_1, \Phi_2, \gamma\rangle$ for which, all of the following hold:*

1. *There is* exactly *one outcome, $v = (v_1, v_2)$ belonging to $NE(G)$ and such that $u_1(v) = u_2(v) = 1$.*
2. *There are initial valuations v_1' for Φ_1 and v_2' for Φ_2 with which the agreed-1-flip protocol will require $\Omega(2^{|\Phi|/2})$ rounds in order to reach this unique equilibrium state.*

PROOF. Immediate from Dunne [5]: initial and final valuations correspond to points on the $|\Phi|$-dimensional hypercube, with the effect of a single negotiation round being to move from the current point to one at (Hamming) distance two from it. The argument in [5] constructs examples where the minimum numer of hyperedges to be traversed in moving from initial to final valuation according to this protocol is bounded below by $(77/128)2^{|\Phi|/2}$. $\square$

6. CONCLUSIONS

Boolean games lie at the intersection of logic, game theory, and computer science, and since they were introduced in 2001, they have attracted steadily increasing attention within the multi-agent systems community in particular. However, a standard criticism of Boolean games is that they are computationally complex, which on reflection is not surprising, given that they are, ultimately, games played on propositional logic formulae. If Boolean game are to find wider application, it will surely therefore be necessary to consider possible routes to tractability in Boolean games. In this paper, we have explored two such routes. First, we considered the idea of relaxing the conditions of pure strategy Nash equilibrium so that we only require that no agent can benefit by flipping no more than k variables. We saw that this very natural idea was, for a certain class of Boolean games (where player's goals are represented as k-clause CNF formulae) in fact sufficient to capture all Nash equilibria; we also saw that k-bounded equilibria lead to a smaller search space than pure Nash equilibria in general. We also saw fixed parameter tractability results for k-bounded equilibria. Next, we considered possible classes of Boolean games for which the corresponding game-theoretic questions were computationally tractable. We identified a condition on goal formulae that leads to tractability: if the *range* of all goal formulae lies within a tractable class of Boolean formula, the corresponding decision problems are much simpler.

In terms of related work, a great deal of work in the algorithmic game theory community has addressed the issues of: (i) the complexity of computing equilibria; (ii) approximate solution concepts such as ϵ-Nash equilibria; and (iii) consideration of classes of games for which computation of solution concepts is tractable; see, for example, the references in [3, 12, 4, 15, 8]. However, this body of work differs from the work presented here in that Boolean games have a very distinctive logical form. Moreover, most work on Boolean games (including the present paper) has considered only *pure* Nash equilibria, while mixed equilibria have received most attention in the algorithmic game theory community. This is of course not surprising, given that the existence of mixed Nash equilibria is guaranteed in finite games, while pure Nash equilibria are not guaranteed to exist in many classes of games.

For future work, several issues suggest themselves. First, it would be interesting to consider mixed Nash equilibria in the context of Boolean games; indeed, it is perhaps surprising that this issue has not been considered previously. In particular, it will be interesting to see how the now-famous PPAD results of [4] manifest themselves in Boolean games. Second, since there is clearly a very close relationship between pure Nash equilibria in Boolean games and the SAT problem for propositional logic, it would be interesting to explore this further, and investigate the extent to which SAT solvers (and QBF/QSAT solvers) can be used to find or verify equilibrium outcomes. Finally, as our results with k-bounded equilibria have demonstrated, it is sometimes possible to have approximate solution concepts that correspond to exact solution concepts on Boolean games in which goal formulae are in certain logical normal forms. It would be interesting to consider this issue further, to see whether other approximate solution concepts "correspond" to their exact counterparts on certain classes of games.

7. REFERENCES

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation*. Springer-Verlag: Berlin, Germany, 1999.

[2] E. Bonzon, M.-C. Lagasquie, J. Lang, and B. Zanuttini. Boolean games revisited. In *Proceedings of the Seventeenth European Conference on Artificial Intelligence (ECAI-2006)*, Riva del Garda, Italy, 2006.

[3] V. Conitzer and T. Sandholm. Complexity results about Nash equilibria. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 765–771, Acapulco, Mexico, 2003.

[4] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a Nash equilibrium. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC-2006)*, Seattle, WA, 2006.

[5] P. E. Dunne. Extremal behaviour in multiagent contract negotiation. *Journal of AI Research*, 23:41–78, 2004.

[6] P. E. Dunne, S. Kraus, W. van der Hoek, and M. Wooldridge. Cooperative boolean games. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2008)*, Estoril, Portugal, 2008.

[7] U. Endriss, S. Kraus, J. Lang, and M. Wooldridge. Designing incentives for boolean games. In *Proceedings of the Tenth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2011)*, Taipei, Taiwan, 2011.

[8] G. Gottlob, G. Greco, and F. Scarcello. Pure nash equilibria: Hard and easy games. *Journal of AI Research*, 24:357–406, 2005.

[9] P. Harrenstein, W. van der Hoek, J.-J.Ch. Meyer, and C. Witteveen. Boolean games. In J. van Benthem, editor, *Proceeding of the Eighth Conference on Theoretical Aspects of Rationality and Knowledge (TARK VIII)*, pages 287–298, Siena, Italy, 2001.

[10] S. Ieong and Y. Shoham. Marginal contribution nets: A compact representation scheme for coalitional games. In *Proceedings of the Sixth ACM Conference on Electronic Commerce (EC'05)*, Vancouver, Canada, 2005.

[11] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley: Reading, MA, 1994.

[12] R. Savani and B. von Stengel. Exponentially many steps for finding a Nash equilibrium in a bimatrix game. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science (FOCS-2004)*, Rome, Italy, 2004.

[13] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press: Cambridge, England, 2008.

[14] S. Szeider. The parameterized complexity of k-flip local search for SAT and MAX SAT. *Discrete Optimization*, 8:139–145, 2011.

[15] B. von Stengel. Equilibrium computation for two-player games in strategic and extensive forms. In N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*, pages 53–78. Cambridge University Press: Cambridge, England, 2007.

A Framework for Modeling Population Strategies by Depth of Reasoning

Michael Wunder
Rutgers University
mwunder@cs.rutgers.edu

Michael Kaisers
Maastricht University
michael.kaisers@maastrichtuniversity.nl

John Robert Yaros
Rutgers University
yaros@cs.rutgers.edu

Michael Littman
Rutgers University
mlittman@cs.rutgers.edu

ABSTRACT

This article presents a population-based cognitive hierarchy model that can be used to estimate the reasoning depth and sophistication of a collection of opponents' strategies from observed behavior in repeated games. This framework provides a compact representation of a distribution of complicated strategies by reducing them to a small number of parameters. This estimated population model can be then used to compute a best response to the observed distribution over these parameters. As such, it provides a basis for building improved strategies given a history of observations of the community of agents. Results show that this model predicts and explains the winning strategies in the recent 2011 Lemonade Stand Game competition, where eight algorithms were pitted against each other. The Lemonade Stand Game is a three-player game with simple rules that includes both cooperative and competitive elements. Despite its apparent simplicity, the fact that success depends crucially on what other players do gives rise to complex interaction patterns, which our new framework captures well.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

Keywords

Iterated Reasoning, Learning in populations, Multiagent Learning

1. INTRODUCTION

The essential problem in multiagent learning has been to apply lessons and techniques from the well-developed field of machine learning to dynamic environments where other decision makers are present. In addition to the obvious challenges of non-stationarity and adversarial adaptation, we might also consider worlds where agents remain anonymous, for one reason or another, such that we may not know at any given moment who we are playing against. This objective, which can be found in such diverse settings as financial markets and politics, raises questions like how to encode a group of complicated strategies compactly so that we may know how to preempt them when a new situation arises. To put the problem in a machine-learning context, we would like to assemble the right feature set to support learning.

Game theory has been built around the concept of Nash equilibria. On the one hand, some games have no unique equilibrium and in many others it is not efficiently computable. On the other hand, if an agent's success primarily depends on others' actions, equilibria may be ubiquitous and completely lose their interpretation as a solution concept. As a result, researchers have developed an alternative theory which has subsequently been proven relevant to human behavior empirically by a wide array of behavioral experiments. This new class of models goes by many names, but the basic idea is that strategies can be classified according to a cognitive hierarchy (CH) with non-reasoning behavior at the bottom and progressively more strategic reasoning at higher levels [4, 5]. There are several hypothesis why this phenomenon occurs. Perhaps people wish to put a minimal amount of effort to the task, or are limited in capabilities. Maybe everyday experience gives them a reasonably good model of what to expect, and this model leads to the behavior. The important thing to note is that this finding appears to be universal, across different cultures and personal backgrounds and over many different games [3].

This paper utilizes and extends an approach that has been applied successfully to behavioral game theory data in the single-shot case [13]. The contribution of this line of previous work was to evalutate a range of models by determining how well they predicted unseen behaviors in matrix-game experiments played by people. This work parallels and builds upon the prior frameworks in several ways but diverges in others. First, we are aligned in our goal of predicting behavior given a data set of obervations, rather than merely explaining the observed behavior. Another shared focus is on the quantal level-k model which has the same properties we would expect to see in our case. While we are primarily interested in repeated settings, unlike the earlier work, there are still similarities to the single-shot case that motivate this approach. For instance, the initial action selection problem can be viewed as a single-shot game given that there is no history, especially if the start to a game significantly influences the course of the succeeding actions. We propose that certain games, like the example case we examine in detail, consist of a series of state-based decisions that closely resemble single-shot games. One of our main contributions is

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

to develop novel statistical methods that can be applied to sequential decisions. While we are studying games played by software agents rather than humans, we maintain that reasoning agents - both programs and humans - are driven by the same behavioral interaction patterns and conclusions of our experiments transfer to human behavior. One benefit to analyzing simulations is the ease, speed, and low cost of running experiments to assemble massive amounts of reproducable data.

To make the investigation more concrete, our game of choice is the Lemonade-stand Game (LSG), which consists of simple rules that illustrate an inherent tension between cooperation and competition [15]. This game is composed of three "lemonade vendors" who compete to serve the most customers in their vicinity, knowing that their competitors are looking to do the same. It is therefore a special case of a Hotelling game, which has been well studied in the economics literature [7, 10]. From a practical standpoint, location games like the LSG have obvious applications for retail establishments in a physical space, but they can also be applied in abstract spaces like on the web or social networks. Hotelling games have another interesting property, which is that they have a price of anarchy equal to $(2n-2)/n$ where n is the number of players and the social optimum is considered to be the minimum score of the n players [1]. Therefore, if many Nash equilibria exist, the resulting payoffs of an equilibrium can severely disadvantage any given player, which makes it all the more urgent for individuals to act according to an accurate population model in these settings.

In previous iterations of the tournament, the customers were spread evenly around the circular beach that functions as the environment and action set. This past year featured a new twist where the thirsty lemonade seekers are distributed unevenly, which creates a new payoff function and strategic challenge every time the game is played. One advantage of using this game as a testbed for studying new multiagent learning methods is that an annual tournament of submitted agents has been run in tandem with the Trading Agent Competition for the past two years. Because of this tournament, there is now a runnable library of agents that can be used for data collection, analysis, and model-building. The upcoming competition adds yet another twist, where the agents will be given the opportunity to learn over many matches (whereas before, the memory ended after one match of 100 rounds). Therefore, we would like a framework that prepares for this lifelong learning problem, while keeping data management to a minimum. This paper will present the CH model as one way to approach such a topic, which has led to an agent that won the 2011 LSG competition [15]. The results of this competition are given in Table 1.

The next section will discuss some of the most popular non-equilibrium focused behavioral models and gives details about the philosophy behind level-based reasoning. Section 3 presents a new framework for generating levels of reasoning in multiplayer games and learning the parameters of this model from observations. In Section 4, we present the Generalized Lemonade-stand Game, which has several properties that make it an intriguing test case for our extended model, and, using actual submitted agents from a LSG tournament, we perform experimental analysis in Section 5. Finally, we close with a discussion in Section 6.

Place	Agent	Score	Std. Dev.
1	Rutgers (our agent)	50.397	± 0.022
2	Harvard	48.995	± 0.020
3	Alberta	48.815	± 0.022
4	Brown	48.760	± 0.023
5	Pujara	47.883	± 0.020
6	BMJoe	47.242	± 0.021
7	Chapman	45.943	± 0.019
8	GATech	45.271	± 0.021

Table 1: Official results of the 2011 Lemonade-stand Game Tournament. Our agent from Rutgers employs a version of the state-based Iterated Best Response (IBR) and won with a significant margin.

2. BACKGROUND

Here, we present the models of behavior that inform our extended model. We should note that some of these models were developed primarily for two-player single-shot games in mind, and therefore need some adaptation for games with more players or that are played repeatedly.

2.1 Quantal Response

One proposed model takes into account the observation that decision makers choose actions according to some function of their value, as opposed to picking the best one. Under this procedure, agents' actions are seen to include some amount of error that can be quantified by the specified function. The most popular function goes by many names, including softmax, exponential weighting, Boltzmann exploration (in the reinforcement-learning literature) or logit quantal response (in game theory).

The quantal decision rule is as follows: A logit quantal action for an estimated utility function u_i of agent i's action a_i results in mixed strategy π_i given by

$$\pi(a_i) = \frac{e^{\lambda u_i(a_i)}}{\sum_{a_i'} e^{\lambda u_i(a_i')}}$$

where λ is defined as the exponential weighting parameter that decides how much error is expected depending on the relative action values.

In games, the utilities depend on opponent strategies, which are not specified by this model. The precision λ is not specified either, and must be set or fit by the model designer. The quantal response mechanism does provide a convenient way to map values into actions that can be used to respond to opposing strategies.

2.2 Level-k

The basis for iterated reasoning models is the idea that agents perform various degrees of strategic reasoning. In this model, dubbed the level-k model, strategies are formed in response to prior strategies known to implement some fixed reasoning capacity [5]. To be specific, an agent acting at level k picks the best response to the strategy at the previous level. In the base case of level 0, the strategy is typically defined as a uniform random action, to provide the reasoning a base that does not perform any reasoning at all. This assumption can be justified by saying that if strategies resulting from this method cannot outperform random action selection, then they are probably not good in any real sense.

Given a set of actions A, Kronecker delta function $\delta : A \times A \rightarrow \{0,1\}$ and utility function $U : A_i \times A_{\neg i} \rightarrow \mathcal{R}$ mapping both agent i's action and the strategies of rest of the population $\neg i$ to real values, the noiseless level-k strategy π_i^k of agent i is

$$\begin{aligned}\pi_i^0(a_i) &= \frac{1}{|A|}\\ \pi_i^k(a_i) &= \delta(a_i, \arg\max_a u(a_i, \pi_{\neg i}^{k-1})).\end{aligned}$$

2.3 Quantal Level-k

This model combines the behaviors of the previous two models to arrive at a strategy calculation that incorporates the recursive best response of level-k with the error robustness of quantal response. The quantal action selection operates on the values derived from the level-k model at the desired setting of k:

$$\begin{aligned}\pi_i^0(a_i) &= \frac{1}{|A|}\\ \pi_i^k(a_i) &= \frac{e^{\lambda u(a_i, \pi_{\neg i}^{k-1})}}{\sum_{a_i'} e^{\lambda u(a_i', \pi_{\neg i}^{k-1})}}.\end{aligned}$$

2.4 Cognitive Hierarchy

We mention this alternative model to address a possible criticism of the level-k model, which is that the best response step essentially ignores levels lower than $k-1$. This crucial point can lead to unwelcome phenomena such as repeating the mistakes of the past. For this reason the cognitive hierarchy model aims to respond to a distribution over previous strategies. Of course, model designers then face a new problem of how to identify the cumulative distribution over levels. The going standard for researchers is the Poisson distribution, which has the elegant property of derivation from a single parameter, τ, that happens to coincide with the average level in the population [4].

If P represents the Poisson function for some τ, then let us define the Poisson-based Cognitive Hierarchy as

$$\begin{aligned}\pi_i^0(a_i) &= \frac{1}{|A|}\\ \pi_i^k(a_i) &= \delta\left(a_i, \arg\max_a \left(\sum_{\kappa=0}^{k-1} P(\kappa) u(a_{\neg i}, \pi_i^{\kappa-1}(a_{\neg i}))\right)\right).\end{aligned}$$

2.5 Algorithms for Modeling Agents

The recursive modeling framework has also had a big impact on the intersection between artificial intelligence and game theory. Recent formal models of reasoning paired with statistical inference include networks of multiagent influence diagrams [8] and interactive Partially Observable Markov Decision Processes [9]. These direct modeling algorithms work best and most efficiently against a single opponent, and can be used in repeated settings. Adding more players to the model adds a great deal of computational complexity, as the interactions proliferate and cris-crossing recursions appear. In contrast, the simpler behavioral economics models are well suited to groups of agents, where specific opponent modeling is not possible. The next section illuminates a process for adapting the idea of Iterated Best Response (IBR) for repeated games against more than a single agent.

	Swerve	Straight
Swerve	3, 3	1, 4
Straight	4, 1	0, 0

Table 2: The game of Chicken.

3. A STATE-BASED IBR FRAMEWORK

Consider the game of Chicken, for which the payoff bimatrix is given in Table 2. The payoffs of the game are such that a player is rewarded for playing the same action again and again because doing so encourages others to change their behavior to the fixed player's benefit (although the converse is also true). In such situations, the actions played by the participants take on the appearance of state, in which the players view the game both as a static environment in which reward must be maximized as well as a Chicken-like scenario where they might prefer to wait for others to back down. The initial positioning of the agents is often of supreme importance in these cases. We will use position or location and action interchangeably given the semi-fixed nature of the actions and for other reasons that will become clear as we develop the concrete example.

We next propose a model for repeated play that addresses these dual goals from the level-based perspective described previously. The decision-making process contains two phases: the initial action-selection problem, resembling a single-shot game with no history, and a state-based decision phase, which begins on the second round and continues until the end. The two phases resemble each other in significant ways, but differ in others. The first round is a special case of the state-based decision because there is no state information that exists; hence it is natural to use the typical assumption that others play randomly so that there is a way to initialize the strategic element. Because the initial action sets the stage for the remainder of the game, choosing a good start should not be ignored. The relationship between first and subsequent rounds is key to successfully identifying good opening moves.

3.1 Initial Action Levels

Although the approach we outline here parallels the one taken by previous game-theoretic models of behavior, we will diverge somewhat to handle cases with more than two agents. The level-k model is successful in cases with only two players, call them Alex and Bill, because if Alex knows that Bill is playing level $k-1$, then of course it makes sense to respond with level k. However, what happens when Carla enters the game? Should the level-k computation be optimized against two level $k-1$ players, or perhaps one level $k-1$ and one level 0 or something else entirely? We do not attempt to answer such questions here, but we would like to make a point about a certain class of games.

Consider a simple one-shot game with three players (Alex, Bill, and Carla) and three actions (1, 2, and 3) on a number line, so that 2 is connected to 1 and 3, but 1 is not connected to 3. The utilties of such a game for player i are defined as

$$U(a_i) = I(a_i)/\#(a_i) + d(a_i, a_{\neg i})$$

where I is the identity function, $\#(a_i)$ is the number of agents playing action a_i, and d is a distance function that returns the minimum distance to the "closest" player in the action space identified by the numbers (i.e. $d(a_i, a_{\neg i}) = \min_{j|j \neq i}(|a_i - a_j|)$). If Carla were to calculate a strategy using level-based reasoning starting at random L0, she would find that L1 should play 3. At L2, the action choice depends on how Carla picks the likely population of Alex and Bill. If Carla believes there are two L1s playing action 3, then she has no preference over action 1 or 2 as they both get utility 3, as long as she does not pick 3 with a guaranteed score of 1. However, if she believes that Alex is an L1 but Bill is an L0, then she has a different choice. In this case, action 1 is worth $(0.5+2+3)/3 = 11/6$, action 2 is worth $(3+1+3)/3 = 7/3$, and action 3 is worth $(1.5 + 1.5 + 1)/3 = 4/3$. In either case, action 2 is optimal (for level 2). In the first case, the double L1 assumption caused Carla to completely misread the structure of the game, and ignore the crucial difference between action 1 and action 2, which is that action 2 has a higher base score.

Games like this one, such as certain location games, can mislead players into dismissing such differences if the lower strategies overlap too much. Therefore, caution dictates that diversity should trump purity when it comes to selecting the lower levels to optimize over. Sometimes it might work to use a distribution over these potential outcomes, but in other instances it may not be computationally possible to do so. In these cases, a mix of previous level and sub-previous level seems to balance out these concerns. Notice that this weakness is adequately addressed by modeling the population as a cognitive hierarchy distribution, but the problem of selecting a correct distribution leads to a multitude of possible responses. Using a cognitive hierarchy model raises computational difficulties, especially in repeated games.

3.2 State-based Levels

Once initial actions have been chosen and played, the position of others is known and the game takes on a state element. Now iterated reasoning separates into two paths. One is the opponent strategy given that our reasoner stays in place, and the other is the expected strategy once our reasoner changes its action. The resulting strategies will be useful in deciding whether or not a new action will exceed the existing cost of inaction. We make the distinction between the current action and a new action due to our realization that others are operating under the assumption that the existing state will likely persist in the future, and the reasoning process is thus continuing even though nothing has necessarily changed yet.

Another way of posing this problem is as the interaction between learning and teaching [11]. The two phenomena are linked because while it is beneficial for agents to learn a better course of action in regards to other agents, they must also understand how to influence the behavior of others.

It is in this context that the concept of regret will be useful to us. An algorithm's total regret is defined as the sum of differences of the total reward of the best strategy that can be adopted and the performance of the algorithm being evalutated up to the current point in time. It has been shown that certain adaptive algorithms have asymptotically no regret over a history of play [2], which suggests that no better strategy can be found. In our present model we would like to focus on a more limited definition of short-term regret as a way to balance the immediate realized sub-optimal past with a hoped-for improved future. Let S denote the state, which is the set of actions chosen by each player. Let us define short-term regret of agent i playing action a_i in state S as follows:

$$\mathcal{R}_{Si} = \max_{a \in A} \sum_{t=t_S}^{T} (u(a) - u(a_i)),$$

where T is the current time and t_S is the first timestep where the game was in the current state.

While an agent can easily minimize this regret by choosing its best response, it also needs to anticipate the reactions of its opponents, which is the purpose of the level-k model. The current state provides a basis for the level computation if we assume that level 0 remains fixed with some unknown probability and otherwise acts randomly. In effect, this assumption leads to the maximizing best response to current state as level 1 as this response is the optimal move unless L0 happens to randomize. If we expect others to execute level 1 in a hypothetical changed state, this expected reaction allows a level 2 agent to compute the action that best prepares for that eventuality. At L2, agents act in a way that maximizes utility when the others eventually best respond. If this expectation is for some reason not met, it may result in inferior performance in the meantime.

A more sophistictated strategy type will aim to limit its vulnerability to this kind of outmaneuvering, in a way that is qualitatively different from simple best response. As such we will take the third level of reasoning to mean an equilibrium strategy. Putting a ceiling on the reasoning process is consistent with the earliest versions of recursive reasoning models which equate L3 with Nash behavior [12]. Reaching the final heights of the level-based action-selection does not necessarily complete a model for repeated games, however. In many games, there are no unique equilibrium strategies or outcomes, and players may wish to bring about better ones. In a wide class of games, this path takes the form of teaching other agents through a consistency of play in order to guide them towards desirable ends.

Suppose we are engaged in a repeated game of Chicken (see Table 2) with both players Driving Straight at each other. This situation is clearly sub-optimal for both: the regret-minimizing action, in a pure strategy sense, is for either player to Swerve. However, the reward-maximizing option is for only the opponent to Swerve so that we may enjoy the high reward. At any given moment, the choice comes down to this dilemma: do I try to wait out my opponent, in the hopes of getting the high score? Or do I succumb to the present urge to fold and take the smaller short-term gain? The only reason to wait in this scenario is due to the long-term summation of projected rewards contingent on the other player backing down. By discounting future rewards, the present value is finite regardless of how long the game continues. According to standard theory, the equation for current value V of future fixed rewards $r_t = r$ received over an infinite time period and discounted by γ per time period can be defined as

$$V = \frac{r}{1-\gamma}.$$

There comes a point where the regret overtakes this value for any given discount value γ. For a value-maximizing agent, this point marks the balance between maintaining a low utility position in hopes that another player will relent and relenting oneself. It is reasonable to expect that the current state of the game will persist for as long as it already has, due to the 50% principle, which states that the most likely temporal location for a random length occurence is halfway through the duration. The present accumulated regret serves to estimate the likely value lost during the upcoming time period, traded off against some optimistic future returns once that period ends. At that point the benefits of outwaiting the opponent are no longer as favorable as myopically choosing an action with higher utility than is currently received. From an observer's perspective, once an agent moves, the discount factor for that agent can be computed. One can also use previous observations to learn an ideal setting of the discount for optimal action.

In games like Prisoner's Dilemma, this waiting dilemma does not exist as there is no short-term gain for choosing to cooperate, nor is there a long-term cost for not doing so. Notice also that in this Chicken example the value γ takes on a meta-game connotation. Denote the two players W and L. Assume w.l.o.g. that $\gamma_W > \gamma_L$. Notice that W gains a higher score, but the higher the value of γ_L, the worse both players perform. We can quantify this game by saying that, subtracting out the regret, the net value for the ultimate winner is $\frac{4}{1-\gamma_W} - \frac{4}{1-\gamma_L}$ whereas the loser gets $\frac{1}{1-\gamma_L} - \frac{4}{1-\gamma_L} = -\frac{3}{1-\gamma_L}$ at the moment the loser backs down.

Therefore, we propose the following method for discovering the ideal value of γ given repeated observations of a population of agents in a game of this type. First, calculate R_s, the total regret experienced in the agent's previous state s, as the difference between expected experienced utility and potential utility in a new state s'. Next, calculate $\hat{U}_s$, the projected alternative score advantage received if it stays in state s assuming the model prediction for the other agents holds. We can then set up this inequality to represent the tradeoff between known regret and potential gains at time $t = \tau$, which yields a condition on γ where if this condition is broken, a different action should be taken.

$$\begin{aligned}
R_s &\geq \sum_{t=\tau}^{\infty} \gamma^t \hat{U}_s \\
\frac{R_s}{\hat{U}_s} &\geq \gamma^\tau + \sum_{t=\tau+1}^{\infty} \gamma^t \\
\frac{R_s}{\hat{U}_s} &\geq \gamma^\tau + \gamma \sum_{t=\tau+1}^{\infty} \gamma^{t+1} \\
\frac{R_s}{\hat{U}_s} &\geq \gamma^\tau + \gamma \sum_{u=\tau}^{\infty} \gamma^u \\
\frac{R_s}{\hat{U}_s} &\geq \gamma^\tau + \gamma \frac{R_s}{\hat{U}_s} \\
\frac{\gamma^\tau}{1-\gamma} &\leq \frac{\hat{U}_s}{R_s}
\end{aligned}$$

In practice, the discount factor can be interpreted as anything from model uncertainty to probability of events. These elements, once accurately learned, combine to form a model that can be effectively used to simulate behavior in previously unseen payoff settings over many rounds, prior to the first round of an interaction. This powerful predictive tool allows a user to identify the most advantageous position to stake out before others without this capability.

4. THE GENERALIZED LEMONADE-STAND GAME

The Lemonade Stand Game is a three-player game with simple rules, yet it gives rise to complex interaction patterns with cooperative as well as competitive elements. Imagine a sunny island with twelve beaches arranged like the numbers on a clock. Early in the morning three lemonade vendors set up their lemonade stand on one of the beaches in the dark (simultaneously) without knowing where the others will sell on that day. Their profit depends on the number of customers they attract, and each customer simply goes to the nearest lemonade stand (in case of a tie it goes to each nearest stand with equal probability).

Martin Zinkevich hosts a competition on an approximately annually basis, where he allows any participating team to submit a single agent [15]. These submissions make up the population of players in the competition. Each triplet plays the game and submissions are scored according to their average performance. The 2009 and 2010 competition featured a uniform customer distribution over the beach locations. In the most recent 2011 competition, agents competed in the Generalized Lemonade Stand Game, where each beach has one, two or three customers with equal probability. Each customer yields a profit of 6 for the stand that attracts it. In expectation, there is 144 cumulative payoff, and the lemonade stand positions decide how this is divided among the three lemonade vendors. This game is repeated for 100 days with the same customer distribution. An example draw with 150 cumulated payoff is given in Figure 1. Based on such a distribution and observations from the days that have passed, the three vendors need to choose their lemonade stand position for the next day. In each of these competitions, simple scripted strategies often outperform complex state-of-the-art online learning algorithms. Note that learning is only allowed within each game and agents are required to completely reset and purge all memory after every match.

4.1 Application of IBR

The Lemonade-Stand Game is a far richer environment than the Chicken or 3-player game described previously, but the same framework applies. This very game has been the

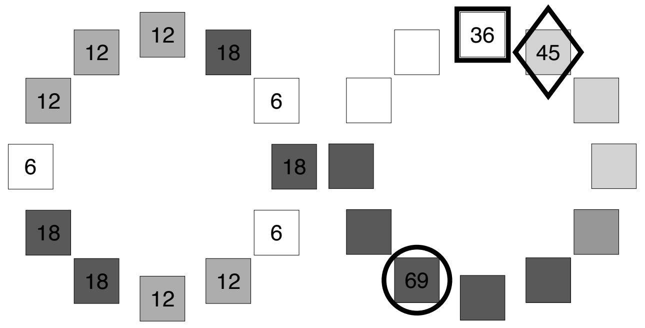


Figure 1: The beaches on Lemonade Island are arranged like the numbers on the clock. Each beach has 1, 2 or 3 customers with equal probability, and each customer gives a profit of 6 (left, darker colors represent higher values). Given lemonade stands ◯, □ and ◊, the payoff to each can be computed (right, colors show to which stand customers go).

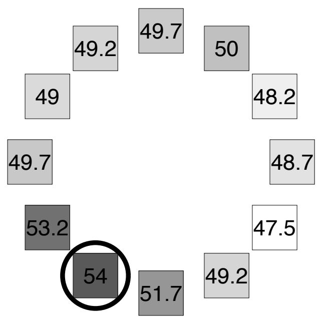


Figure 2: Expected payoffs for reasoning Level 1: darker colors represent higher values. The circle denotes the best available action against two random players (L0).

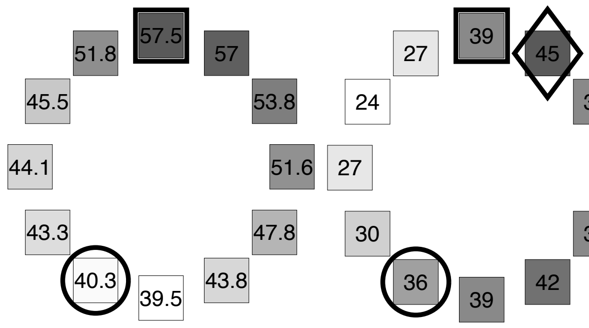


Figure 3: Expected payoffs for reasoning Level 2 (left) and 3 (right): darker colors represent higher values. The □ denotes the best reply to one L1 player ◯ and one uniform random player (L0). The ◊ denotes the best available action against one L1 player ◯ and one L2 player □.

subject of a few articles since the tournament began. Some participants have taken a more general approach to the planning elements [6] while others have addressed the way that iterative methods can produce the behaviors seen [14]. However, in the generalized version of the game, a richer model is required to handle the multitude of cases and viable potential strategies.

We assume an agent at Level 0 does not do any reasoning. Thus, it will chose an arbitrary action at random. An agent of the next reasoning level will assume its opponents both use reasoning Level 0. Such a player of Level 1 can compute its expected payoffs for all possible actions and choose the best reply (see Figure 2).

At all higher levels, the player will compute a best reply to opponents of the previous two levels. That is, a Level 2 player computes the expected payoffs against one Level 0 and one Level 1 opponent. Similarly, a Level 3 player computes the best reply to one Level 1 and one Level 2 opponent (see Figure 3 for an illustration). Structuring the iterative model in this way is a design choice, which is motivated both by the problems with overlapping strategies already mentioned and by the success in capturing the three-player interactions as presented in the following section.

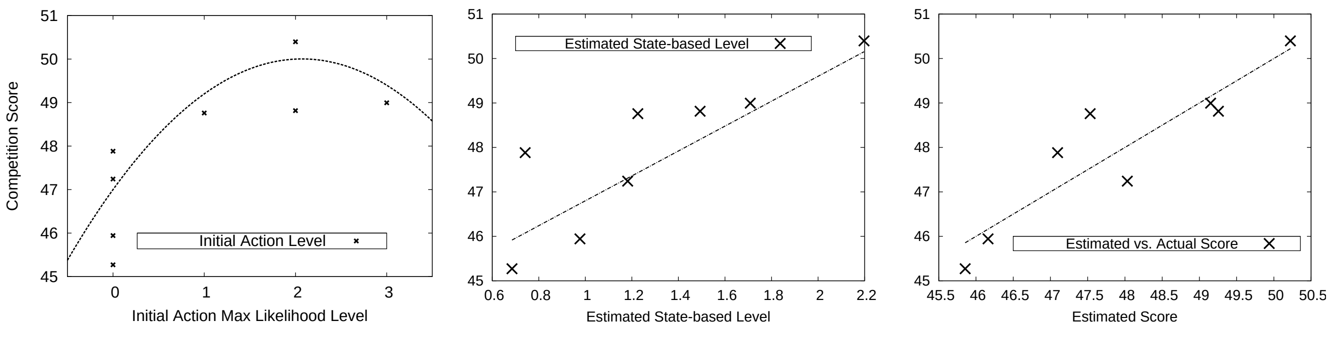
5. EXPERIMENTS AND VALIDATION

The experimental data presented here is derived from the submitted set of agents from the July 2011 Generalized LSG tournament, which implements the LSG with varying customer distributions. We have two goals in the following analysis. First, to show that the features resulting from the derived level-based model accurately predict performance in the tournaments. This is confirmed by the more advanced strategies under this definition being among the winners. Second, to demonstrate how this level data can be utilized to efficiently determine the distribution over types that are present in an ongoing series of games, and subsequently outflank them. This goal is of interest for upcoming competitions, where agents will have the chance to observe opponents in action over many games, albeit anonymously, and respond as they wish. As a side point, it has been observed that in LSG's current form, there is not much opportunity for modeling and responding to opponents within games, as matches are often settled within a small number of moves. Therefore any agent that possesses an accurate model of behaviors will have a great advantage when it comes to planning a strategy in matches with previously unseen density functions.

5.1 Learning Distributions over Levels

The quantal level-k model (Section 2.3) provides a method for producing the strategies at each level given a precision value λ. To produce an estimate of the strategy executed by an agent, observe its action and the probability that each strategy would have selected it. Using this probability we arrive via Bayes at the likelihood that this agent is acting according to this strategy. The normalized likelihoods for all strategies give an estimated model of this agent. As described in the extended model 3, the two sets of observations that determine the distribution over levels are initial actions and new actions. Because the number of these observations may tend to be small in a single instance of the game at hand, we must gather data over many matches, with many different payoff functions. There are several ways to build models from this data. In the comprehensive survey and analysis done with human data [13], the authors used a maximum likelihood method to be able to make predictions of future behaviors. For our purposes, this predictive ability may not be enough because we would like to be able to generate strategies in response to a learned model. Therefore, a compromise solution is to simply average together the resulting likelihoods for every instance observed.

The mark of a successful model is to predict outcomes, and the state-based multiplayer IBR framework presented here is no exception. In this section we test the correlation between the various model parameters and final competition scores. Our results can be summarized in Figure 4. We took the level distributions for each agent and found the average estimated level for both initial action and state action in this manner. Then we examined the history leading up to a change in state and recorded the regrets of the participating agents, and thereby arrived at bounds on γ. This step gives three total model parameters, including the discount factor. All three are highly positively correlated with final performance. If we combine all three values to predict the score, we reach a 0.83 coefficient of correlation and over 0.9 for prediction of rankings. The initial action level has highest performance at L2, on account of other agents playing



(a) (b) (c)

Figure 4: A comparison of the final competition score with the maximum likelihood level ($R^2 = 0.73$), estimated state-based level ($R^2 = 0.72$) and estimated score ($R^2 = 0.83$). (a) The best fit over the data points is a quadratic function because most of the agents are L1 or less, which means that L3 does worse than an L2 strategy in this population. (b) Average likelihoods of each level are computed and normalized, and then each level is weighted by the corresponding likelihood. The Spearman's rank coefficient is 0.93. (c) The estimated score is output by the final model as a function of initial level, state-based level, and discount factor. The Spearman's rank coefficient is 0.9. The trendline shown is X=Y.

sub-L2 strategies. Recall that the level-k model prescribes responding to the average degree of sophistication in the population, and not necessarily the highest possible strategy calculation with the most reasoning.

5.2 Building Strategies from a Learned Level-based Model

Once the strategy distribution is in place, we turn to finding optimal responses to the underlying population. The process used to accomplish this task is basically the mirror image of the learning mechanism. Given a new LSG instance, we discover the initial action levels, up to the third. In our case three is sufficient because the reasoning only needs to handle three agents. It so happens that by choosing one action from each of the three level strategies will yield a very good approximation to the likely final state of a series of best response steps, starting from any initial positions. In the event that the space cannot be divided in this way, there are probably a high number of equally good starting points distributed more or less evenly around the circle. We have done analysis to confirm this point but cannot show it for lack of room. We will suffice to call these actions *stable* for lack of a better word.

Once these three candidate actions are chosen, we can easily find the scores assuming that there is one agent at each location, giving a ranking of these three, which will not necessarily correspond to the level that first produced them. (See Figures 4-6 for a visual example explanation.) It is unlikely that the three players will end up picking a different one of the three stable actions. More likely is that at least one of our two opponents will choose the highest ranked action, as all else equal it would be the best one. Fortunately this situation is ideal for us to analyze using the regret-based approach mentioned in Section 3. If we have an estimated discount factor (γ) for our target population, then we know how long to expect an opponent to wait before switching to the lesser stable action, leaving our agent in command of the best one. If γ is sufficiently low, it will be worth the initial cost to be the lucky agent to collect the higher score. However, if the population has demonstrated high γ and therefore a lot of patience, then it may in fact be optimal to take one of the lower ranked stable actions, and hope that our opponents end up fighting over the highly ranked location. The parameterized model accounts for these opposing forces and combines them to compute the estimated values for each of these stable points over an entire game, prior to the game starting. Although we have described this process in words here, an agent we have built is able to quantitatively discover this solution automatically, and thus fully implement a model-based response.

In Table 3, we show the performance of our agent, the Full-Model agent, against the top four challengers in the latest tournament. The best submissions are used since other strategies have been observed to emulate them after some time [16], and we would like our agent to perform well against likely future competitors. This agent runs the model to estimate the game-length values of the best starting points, and selects the best of these accordingly. Once the game has begun, it switches into state-based mode and makes regret-informed decisions using an internal value of γ that may be adjusted based on observations from the population. Table 4 replicates the original tournament with the addition of the Full-Model agent.

6. CONCLUSION

This article demonstrated a model synthesizing several

Table 3: Results of an internal experimental match-up including the top four agents of the 2011 Tournament and an agent constructed using the full state-based IBR. 100 repetitions of each match shown.

Ranking	Agent	Score	Error
1	Full-Model	51.61	0.039
2	Rutgers	49.36	0.037
3	Alberta	47.63	0.039
4	Harvard	47.60	0.032
5	Brown	43.10	0.034

Table 4: Results of an internal experimental match-up including all agents of the 2011 Tournament and an agent constructed using the full state-based IBR. 100 repetitions of each match shown.

Ranking	Agent	Score	Error
1	Full-Model	50.48	0.042
2	Rutgers	50.06	0.033
3	Harvard	49.21	0.035
4	Alberta	48.83	0.030
5	Brown	48.40	0.036
6	Pujara	47.27	0.039
7	BMJoe	46.95	0.037
8	Chapman	45.46	0.035
9	GATech	45.03	0.031

schools of thought regarding the modeling and prediction of agent behavior, especially including level-based reasoning and regret minimization. We built upon established methods for constructing a hierarchy of strategies through the tried-and-true best response operation as pieces of a robust process and adapted them to operate in new domains. Our present framework has a corresponding automated algorithm that outputs the strategies at each level. Given a data set of behavioral observations, the model infers the relevant parameters and frequencies of the constructed strategies. The final step is to translate the model into useable strategies, as we show through the Lemonade-stand Game example. The strength of our model is its compactness and ability to pre-simulate the likely course of action before the game is populated with agents, giving us a sizeable advantage when learning is allowed over many periods. The LSG tournament provides a case study for this model in action, and its past and present success is credited to the power of an automated procedure based on our framework.

Algorithm 1 Model Pseudocode for LSG

```
Input: $\gamma$ (estimated optimal population discount factor)
Input: $P$ (estimated population level probabilities)
getAction()
  if $turn = 0$ then
    $stable[0] \leftarrow$ highest density location
    $stable[1] \leftarrow$ 2nd highest density location
    $stable[2] \leftarrow$ bestResponse($stable[0], stable[1]$)
    $max \leftarrow$ maxUtil($stable[0], stable[1], stable[2]$)
    $potRgrt \leftarrow$ utility of non-$max$ locations
    $collideRwrd \leftarrow$ utility if two players choose $max$
    $projRwrd \leftarrow (max - collideRwrd)$
    $futVal \leftarrow projRwrd/(1-\gamma)$
    $expRwrd \leftarrow$ comb. of $futVal$ and $projRwrd$ under $P$
    return $stable$ action with highest $expRwrd$
  else
    /* Use State-Based IBR */
    $cumRgrt \leftarrow cumRgrt +$ utilDiff(bestAlt, curConfig)
    $projRwrd \leftarrow$ utilDiff(curAction, newOppConfig)
    if $\frac{\gamma^{turn}}{1-\gamma} cumRgrt > projRwrd$ then
      $cumRgrt \leftarrow 0$
      return bestAlt in bestAltConfig
    else
      return curAction
    end if
  end if
```

7. REFERENCES

[1] A. Blum, M. T. Hajiaghayi, K. Ligett, and A. Roth. Regret minimization and the price of total anarchy. *In Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 373–382, 2008.

[2] M. Bowling. Convergence and no-regret in multiagent learning. *Advances in Neural Information Processing Systems 17 (NIPS)*, pages 209–216, 2005.

[3] C. F. Camerer. *Behavioral Game Theory: Experiments in Strategic Interaction.* Princeton University Press, 2003.

[4] C. F. Camerer, T.-H. Ho, and J.-K. Chong. A cognitive hierarchy model of games. *Quarterly Journal of Economics*, 119:861–898, 2004.

[5] M. Costa-Gomes, V. Crawford, and B. Broseta. Cognition and behavior in normal-form games: An experimental study. *Econometrica*, 69(5):1193–1235, 2001.

[6] E. M. de Côte, A. Chapman, A. M. Sykulski, and N. R. Jennings. Automated planning in adversarial repeated games. *UAI*, 2010.

[7] J. J. Gabszewicz and J.-F. Thisse. Location. *Handbook of Game Theory with Economic Applications*, 1992.

[8] Y. Gal and A. Pfeffer. Networks of influence diagrams: Reasoning about agents' beliefs and decision-making processes. *Journal of Artificial Intelligence Research (JAIR)*, 2008.

[9] P. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multiagent settings. *Journal of AI Research (JAIR)*, 24:49–79, 2005.

[10] H. Hotelling. Stability in competition. *The Economic Journal*, 39:41Ű57, 1929.

[11] K. Leyton-Brown and Y. Shoham. *Multiagent Systems: Algorithmic, Game Theoretic and Logical Foundations.* Cambridge University Press, 2009.

[12] D. O. Stahl and P. W. Wilson. On players' models of other players: Theory and experimental evidence. *Games and Economic Behavior*, pages 218–254, 1995.

[13] J. R. Wright and K. Leyton-Brown. Beyond equilibrium: Predicting human behavior in normal form games. *The Twenty-Fourth Conference on Artificial Intelligence (AAAI-10)*, 2010.

[14] M. Wunder, M. Kaisers, M. Littman, and J. R. Yaros. Using iterated reasoning to predict opponent strategies. *International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2011.

[15] M. Zinkevich. The lemonade game competition. http://tech.groups.yahoo.com/group/lemonadegame/, December 2009.

[16] M. Zinkevich, M. Bowling, and M. Wunder. The lemonade stand game competition: Solving unsolvable games. *ACM SIGecom Exchanges*, 10, 2011.

Detection of Suspicious Behavior from a Sparse Set of Multiagent Interactions

Boštjan Kaluža
Jozef Stefan Institute
Ljubljana, Slovenia
bostjan.kaluza@ijs.si

Gal A. Kaminka
Bar Ilan University
Ramat Gan, Israel
galk@cs.biu.ac.il

Milind Tambe
University of Southern California
Los Angeles, California
tambe@usc.edu

ABSTRACT

In many multiagent domains, no single observation event is sufficient to determine that the behavior of individuals is suspicious. Instead, suspiciousness must be inferred from a combination of multiple events, where events refer to the individual's interactions with other individuals. Hence, a detection system must employ a detector that combines evidence from multiple events, in contrast to most previous work, which focuses on the detection of a single, clearly suspicious event. This paper proposes a two-step detection system, where it first detects trigger events from multiagent interactions, and then combines the evidence to provide a degree of suspicion. The paper provides three key contributions: (i) proposes a novel detector that generalizes a utility-based plan recognition with arbitrary utility functions, (ii) specifies conditions that any reasonable detector should satisfy, and (iii) analyzes three detectors and compares them with the proposed approach. The results on a simulated airport domain and a dangerous-driver domain show that our new algorithm outperforms other approaches in several settings.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: intelligent agents, multiagent systems

General Terms

Algorithms, Security, Experimentation

Keywords

suspicious behavior, multiagent interactions, scoring functions

1. INTRODUCTION

There is a significant amount of research in suspicious activity detection, given its importance in many domains [1, 5, 9, 16]. The goal is to augment the traditional security measures by scrutinizing the behavior of all the subjects in the environment. We target a large class of applications where no single event is sufficient to make a decision about whether behavior is suspicious or not. Instead, we face a sparse set of *trigger events* that identify interesting parts characterizing the behavior trace. Examples include a potentially suspicious passenger who appears to turn away in the presence of security personnel, but not blatantly so, hence no single such event is enough to raise suspicion. The main question we address is how to combine multiple events to decide whether an event trace corresponds to the behavior of a normal or a suspicious person.

There are four challenges that need to be addressed. First, there is no one significant event or incident that would help us to immediately reach a decision; a series of trigger events allows us to reach a decision. Second, we have no knowledge about the exact plans devised by a suspicious person. Third, trigger events include the interactions of multiple agents making recognition in the presence of noise difficult. Fourth, the degree of suspiciousness contributed by a suspicious event depends on the agent's behavior in the past. For example, the third suspicious event is evaluated differently than the first, since the agent's previous behavior indicates a tendency to behave suspiciously. Hence, the simple counting of suspicious trigger events cannot be applied, since it accumulates all the events linearly. Furthermore, most of the plan-recognition methods, which rely on a plan library, are insufficient, since plans are not known in advance.

This paper presents a two-step approach to suspicious behavior detection from a sequence of an agent's actions. The first step detects trigger events, i.e., interesting parts of the sequence that serve as evidence, and estimates the probability that an event is suspicious. For this task we present an approach using coupled hidden Markov models [4] that are able to model interactive behavior. The second step combines evidence from multiple events in order to determine suspiciousness.

The key contributions of this paper are in the second step, which is defined as a decision problem: Is the behavior of an agent suspicious given a sequence of trigger events? First, we formally describe the detection problem and specify the conditions that any reasonable detector should satisfy. Second, we analyze three detectors, namely the naive Bayes detector, the hidden Markov models and the utility-based plan recognition (UPR). These detectors, however, either simplify the problem or evaluate the events linearly. Finally, we present a novel detector that is a generalization of UPR and denoted as Function-UPR (F-UPR): (i) we define utilities as a set of functions over state transitions and observations; and (ii) we introduce an observation utility function that is especially suitable for suspicious behavior detection, since it is able to evaluate events non-linearly. The experimental evaluation on a simulated airport domain first compares the three detectors with our proposed approach. The best two approaches are additionally compared on the dangerous-driver domain.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

2. MOTIVATING DOMAIN AND RELATED WORK

Airports require numerous security solutions, including the identification of suspicious activities among passengers and staff in surrounding areas. Our goal is to monitor passengers during the time they spend at the airport and to detect those that indicate a high level of stress, fear or deception. It is reasonable to assume that there is a camera network to track a passenger throughout the airport. We focus on a task where no single event is sufficient to identify a suspicious passenger, but a series of events establishes the decision over time. The detection of events might be limited due to noise or an inability to extract some features (e.g., using a ceiling-mounted camera one can extract the trajectory of a passenger, but not facial expressions), hence a normal person may appear suspicious (and vice versa). Also, a precise plan of the suspicious passenger is not known in advance. Other domains of interest may include catching a reckless driver executing dangerous (but still legal) maneuvers [2], detecting a pirate vessel that plans to capture a transport vessel and therefore avoids security patrols, etc.

There are two approaches to detecting deviant behavior [2]: *suspicious* and *anomalous* behavior detection. The first approach assumes a behavior library that encodes *negative behavior*, and thus recognizing observed behavior corresponds to identifying a match in the library. The second approach uses the behavior library in an inverse fashion, meaning that the library encodes only *positive behavior*. When an observed behavior cannot be matched against the library it is considered as anomalous. Several approaches have been proposed to tackle the problem either way. In the airport scenario various systems were introduced to automatically detect some of the threats, such as leaving objects behind [10], suspicious trajectory paths [16], thefts [10], and vandalism acts and fights [12]. There is also a commercially available system [7] that is able to detect events such as running passengers, climbing over a fence, etc. However, these approaches mainly deal with the detection of single incidents, which are clearly suspicious. They do not address accumulating suspicion as we do.

Another area of related work includes hidden Markov models (HMMs) [13] that are widely used in traditional activity recognition for modeling a sequence of actions. Brand et al. [4] introduced coupled HMMs as an extension with multiple hidden interacting chains that are able to model interactive behavior. Duong et al. [5] focused on the duration of activities and introduced switching hidden semi-Markov models that provide probabilistic constraints over the duration of plans, and applied them to the detection of anomalies in the activities of daily living. Although widely used, HMMs may become inadequate when actions are more complex or have long-term temporal dependencies [11].

Plan recognition algorithms may use a hybrid approach for suspicious activity recognition. A symbolic plan recognizer is used to filter consistent hypotheses, passing them to an evaluation engine, which focuses on ranking. Geib and Goldman presented PHATT [8], a probabilistic approach based on tree grammars able to cope with interleaved goals, partially ordered plans, and failed observed actions. Sukthankar and Sycara [14] addressed plan recognition for multiagent teams, where plans were ordered by linear accumulation of observed actions consistent with the plan. Another approach is presented by Avrahami-Zilberbrand and Kaminka [2, 3]. Utility-based Plan Recognition (UPR) introduces utility to the observer in selecting the recognition hypotheses. The main strength of UPR is that it can incorporate an observer's bias to events with a low likelihood, for example, the a-priori probability for planting a bomb is very low, but detecting it has a high expected utility. We further discuss this approach in Section 5.3.

Furthermore, intrusion detection systems analyze a variety of user activities to identify suspicious computer activities. Helman and Liepins [9] proposed an intrusion detection system that provides a rating for computer activities, demonstrating frequency estimator and matching rules. Esponda et al. [6] analyzed tradeoffs between positive and negative activity patterns in the library and presented an approach based on partially matching rules. These approaches similarly address the problem of how to decide whether a user's activity is suspicious, but differ significantly in using a different approach to match and assess behavior.

3. DEFINITIONS AND ASSUMPTIONS

Our methods are general, but for illustrative purposes we will make use of the airport domain to provide examples. We treat subjects as agents in a multi-agent environment. At this point we assume that we can perfectly observe their actions.

Definition 1. Action a_t is a tuple of observed feature values $\langle f_1,...,f_n \rangle$ that describe state of an agent at a given time stamp t.

Definition 2. Action trace $\mathbf{a}^{(l)}$ is a totally-ordered sequence of l actions $\mathbf{a}^{(l)} = (a_1, a_2, ..., a_l)$.

Definition 3. Trigger event $x^{i,j} = (a_i,...,a_j)$ is a subsequence of action trace $\mathbf{a}^{(k)}$ (s.t. $1 \leq i < j \leq k$). A trigger event x is described by probabilities that the corresponding subsequence is suspicious $s(x)$ and normal $n(x)$.

Definition 4. Event trace $\mathbf{x}^{(k)}$ is a totally-ordered sequence of k trigger events $\mathbf{x}^{(k)} = (x_1, x_2, ..., x_k)$.

We address the problem of suspicious behavior detection in two steps, as shown in Figure 1. The first step analyzes an action trace and the surrounding environment to detect trigger events that characterize its interesting parts. The event trace then enters the second step, where it is evaluated. If the evaluation result exceeds a threshold value or is large relative to other evaluations of the event traces, then it is considered as suspicious.

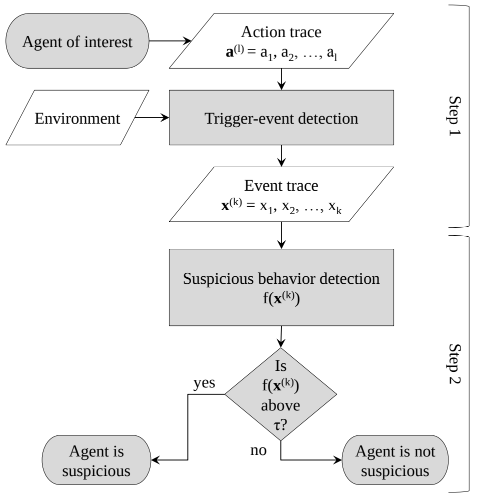


Figure 1: Two-step detection of suspicious behavior: (1) detection of trigger events and (2) detection of suspicious behavior.

Trigger events can be any kind of partial observations we are able to extract from the domain. In the airport domain, one can focus on people exhibiting indications of suspicious behavior, such as taking photos of critical infrastructure, revisiting the same location, evading the area when noticed, standing in customer service but not requesting the service, etc. We focus on a well-known detector obtained from conversations with domain experts. We observe the interactions between agents at the airport, more precisely, we are interested in how a passenger behaves in the presence of a uniformed authority figure. A person exposed to a high level of stress produces behavior that indicates fear, anxiety, pressure, tension, deception, etc. Hence, it is rational for the suspicious agent to minimize contacts with the authorities. Note, that no single avoidance is enough to raise a flag, but many such events put together cause the person to be treated as suspicious.

A trigger-event detection able to identify interactive behavior may rely on coupled hidden Markov models (CHMMs), which are briefly described below. The reader is referred to [4] for details; the CHMMs are not the main contribution of the paper. The observations consist of two action traces, namely the action trace of the agent of interest and the action trace of an authority agent when they are within some predefined radius. The CHMMs are able to model the complex, interactive behavior by two HMM chains, where the hidden states from one chain directly impact on the hidden states from the other chain. Figure 2 illustrates the CHMM for a pair of action traces with length $l = 3$. The current state Q_t^A of agent A is affected by both its previous state Q_{t-1}^A and previous state Q_{t-1}^B of the agent B (similarly Q_t^B is affected by Q_{t-1}^B and Q_{t-1}^A). Each state Q_i also impacts the corresponding observation state Y_t. For example, if the authority agent moves toward the suspicious agent, the next state of the latter takes this into account and produces an action for an avoidance maneuver.

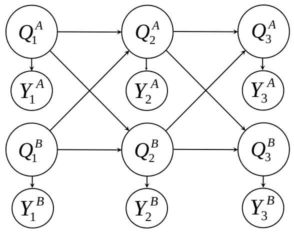


Figure 2: An example of CHMM for a pair of action traces with length $l = 3$.

A regular passenger may not turn or do anything different in the presence of authorities, while a suspicious person will (although as described below, an observer may not have perfect observability). Therefore, we create and train two CHMMs: $\hat{N}_I$ models the interactions produced by authorities and regular passengers, while $\hat{S}_I$ models the interactions produced by authorities and suspicious passengers. For a new event (interaction) x we compute the posterior probability that the event is generated with both models yielding $\hat{n}_I(x) = Pr\{x|\hat{N}\}$ and $\hat{s}_I(x) = Pr\{x|\hat{S}\}$, respectively.

4. PROBLEM DEFINITION

This section formally analyzes how to evaluate a sequence of trigger events. We leverage the Bayesian framework for intrusion detection [9] for the problem definition. At each time step t we observe an event x_t, generated by a hidden stochastic process H. Now suppose that H is a mixture of two auxiliary stochastic processes, namely the normal process N and the suspicious process S that correspond to a normal and a suspicious passenger. The random variable $y_t = 0$ if x_t is generated by N and $y_t = 1$ if x_t is generated by S. Since a suspicious passenger always emits a suspicious event (and a normal person a normal event), y for a specific agent does not change over time. In reality, there can be many subprocesses contributing to each of N and S, i.e., many normal users with different behavior patterns; however, here we assume only a single N and a single S that capture all the variability.

To this point we assumed that an observer is able to perfectly observe whether an event is generated by S or N. In practice, however, it may appear that a normal person emits suspicious events (or vice-versa). An observer might be limited for various reasons, such as an inability to detect characterizing features and noisy trigger-event detectors. Therefore, we relax this assumption as follows. An event x_t is observed as generated by N with the probability $n(x_t) = Pr\{H(t) = x_t|y_t = 0\}$ and as generated by S with the probability $s(x_t) = Pr\{H(t) = x_t|y_t = 1\} = 1 - n(x_t)$. The mixture distribution of an event x_t and a prior probability λ is

$$Pr\{H(t) = x_t\} = \lambda s(x_t) + (1-\lambda)n(x_t). \quad (1)$$

The objective of suspicious behavior detection is to identify those traces $\mathbf{x}^{(k)} = (x_1, x_2, ..., x_k)$ that are likely to be suspicious activities, i.e., traces $\mathbf{x}$ for which

$$Pr\{y = 1|H(t) = x_t, t = 1, ..., k\} > \tau, \quad (2)$$

is above some threshold τ or is large relative to the probability for other traces.

The reason why this problem is difficult is because of the non-linear effect. Consider the following example. Suppose we observe a person do a U-turn in front of a police officer, so that the likelihood that this was a suspicious person becomes high. Later we see the same person doing a half-turn in front of a police officer. This trigger event if seen on its own, would not contribute much to the overall suspicion. However, following the initial turn we had observed, this new turn is a much stronger evidence to be attributed to the overall suspicion, because we bias the new event with our previous observation.

Theoretically, it might be possible to optimally detect suspicious behavior using Eq. (2). Unfortunately, this is usually not the case in practice. To see this, let us assume a prior probability $\lambda = Pr\{y_t = 1, t = 1, ..., k\}$. In most cases λ is close to 0, since in real-world applications suspicious activities are rare. Let the stochastic processes N, S and H denote $n(\mathbf{x}^{(k)}) = Pr\{H(t) = x_t, t = 1, ..., k|y = 0\}$, $s(\mathbf{x}^{(k)}) = Pr\{H(t) = x_t, t = 1, ..., k|y = 1\}$, and $h(\mathbf{x}^{(k)}) = Pr\{H(t) = x_t, t = 1, ..., k\}$, respectively. Using Bayes theorem we can derive from Eq. (2)

$$Pr\{y = 1|H(t) = x_t, t = 1, ..., k\} = \frac{\lambda \cdot s(\mathbf{x}^{(k)})}{h(\mathbf{x}^{(k)})} = \quad (3)$$

$$= \frac{\lambda \cdot \prod_{t=1}^{k} s(x_t|x_{i,i=t-1,...,1})}{\lambda \prod_{t=1}^{k} s(x_t|x_{i,i=t-1,...,1}) + (1-\lambda)\prod_{t=1}^{k} n(x_t|x_{i,i=t-1,...,1})}$$

To this point we implicitly assumed that the distributions λ, n and s are reliably estimable. The degree to which this assumption is valid depends on our detection capability. Suppose we have a sufficiently large dataset D_l of labeled event traces, we can estimate the prior probability λ from the D_l using the relative frequency, presenting the number of traces generated by a suspicious agent divided by the total number of traces (since traces can be of different lengths, the quotient is normalized by the traces' length). Note that in order to compute $Pr\{H(t) = x_t, t = 1, ..., k|y = 1\}$ we have to evaluate

$$s(x_1) \cdot s(x_2|x_1) \cdot ... \cdot s(x_k|x_{k-1}, ..., x_1) \quad (4)$$

While some first terms, i.e., $s(x_t), s(x_t|x_{t-1})$, can still be estimated, the estimation of latter terms including increasingly more history becomes less and less reliable. In real-world applications we have no direct knowledge of the values of the conditional probabilities, i.e., we are unable to specify the probability of an event given all the possible combinations of history. For this reason we must approximate the Bayes optimality in general. In particular, we will be concerned with estimating $Pr\{y = 1|H(t) = x_t, t = 1, ..., k\}$ using approximate approaches.

Given an event trace, some events may appear suspicious and some not. Hence, detection systems must have a scoring function that combines the evidence. The output of a function is interpreted as the degree of suspicion attributed to the event trace. Although any two scoring functions need not be exactly the same, we can specify the conditions that any reasonable scoring function must satisfy. The class defined below appears to be both natural and general.

The detection system can employ a *scoring function f* that interprets events to produce a score characterizing the overall suspicion of the trace. Given a threshold value τ and an event trace $\mathbf{x}^{(k)}$ we can classify $\mathbf{x}^{(k)}$ as suspicious if $f(\mathbf{x}^{(k)}) \geq \tau$.

Definition 5. A scoring function f over a trace of events $\mathbf{x}^{(k)}$ is a function

$$f : \bigcup_{k=1}^{K} \mathbf{x}^{(k)} \rightarrow \mathbb{R}$$

The function f assigns a real value to any trace $\mathbf{x}^{(k)}$ of length $k = 1, ..., K$.

Let $\Delta(x_t)$ decide whether a single event x_t is suspicious or not

$$\Delta(x_t) = \begin{cases} 1; & \text{if } s'(x_t) \geq \tau' \\ 0; & \text{else} \end{cases}, \tag{5}$$

$$s'(x_t) = \frac{\lambda \cdot s(x_t)}{\lambda \cdot s(x_t) + (1-\lambda) \cdot n(x_t)}. \tag{6}$$

Definition 6. A class of *well-behaved* functions consist of scoring functions s.t. $\forall \mathbf{x}^{(k)}, x_{k+1}$:

$$\begin{aligned} f(\mathbf{x}^{(k)}, x_{k+1}) \geq f(\mathbf{x}^{(k)}) \qquad & \text{if } \Delta(x_{k+1}) = 1, \\ f(\mathbf{x}^{(k)}, x_{k+1}) \leq f(\mathbf{x}^{(k)}) \qquad & \text{if } \Delta(x_{k+1}) = 0. \end{aligned}$$

The conditions imply that: (i) the scoring function f's evaluation increases when a new suspicious event is added to the trace and (ii) decreases when a normal event is added to the trace. The well-behaved scoring functions are motivated by the key observation that a suspicious event x_{k+1} (i.e., $\Delta(x_{k+1}) = 1$) is more likely to be generated by a suspicious process S than a normal process N, regardless of the history $\mathbf{x}^{(k)}$, i.e.,

$$\begin{aligned} s(x_{k+1}|\mathbf{x}^{(k)}) \geq n(x_{k+1}|\mathbf{x}^{(k)}) \qquad & \text{if } \Delta(x_{k+1}) = 1 \text{ and} \\ s(x_{k+1}|\mathbf{x}^{(k)}) \leq n(x_{k+1}|\mathbf{x}^{(k)}) \qquad & \text{if } \Delta(x_{k+1}) = 0. \end{aligned}$$

5. DETECTORS

In this section we analyze the approaches that decide whether an event trace is suspicious. First, we discuss the naive Bayes detector that relaxes the initial assumptions. Next, we discuss an approach that directly tackles the problem of estimating the likelihood that a trace was generated by a suspicious process using HMMs. Finally, we analyze an approach based on plan recognition and present two extensions: (1) we define utilities as a potential function; and (2) we present an observation utility function able to address non-linear accumulation.

5.1 Naive Bayes Detector

A naive approach assumes that events are independent, which means that the current event depends only on the current time step t and not on the time steps prior to t. The evaluation of Eq. (3) is simplified using the naive assumption:

$$Pr\{y = 1|H(t) = x_t, t = 1, ..., k\} = \frac{\lambda \cdot \prod_{t=1}^{k} \hat{s}(x_t)}{\lambda \cdot \prod_{i=1}^{k} \hat{s}(x_t) + (1-\lambda) \cdot \prod_{i=1}^{k} \hat{n}(x_t)} \tag{7}$$

We have to evaluate the probability $Pr\{H(t) = x_t|y_t\}$ that an event is generated by a normal process $\hat{n}(x_t)$ and a suspicious process $\hat{s}(x_t)$, which is tractable in terms of evaluation. The approaches for estimating $\hat{n}$ and $\hat{s}$ may include a frequentist estimator, hidden Markov models, k-nearest neighbors, neural networks, etc. We showed an approach using CHMM in Section 3. An evaluation of the event trace is also well behaved when $\tau' = \lambda$.

In practice, the assumptions may oversimplify the model; however, we will use it as a baseline in our experiments.

5.2 Hidden Markov Models

An estimation of the conditional probabilities including the history can be encoded with hidden Markov models (HMMs) [13]. A HMM is a temporal probabilistic model with two embedded stochastic processes: an unobservable (hidden) process Q, which can be observed only through another (visible) stochastic process O. Each state in Q has state-transition probabilities (which are visible) and a probability distribution over the possible values of O. The key assumption is that the current hidden state of the agent is affected only by its previous state.

Now suppose we create a HMM to estimate $Pr\{H(t) = x_t|y = 1, t = 1, ..., k\}$, more precisely, it models the probability that a trace of events is generated by a suspicious agent. The hidden states of the process Q may be referred to as internal states presenting the intentions of the suspicious agent. For the sake of clarity, let us assume only two hidden states: a normal intention and a suspicious intention, emitting normal and suspicious events, respectively. The transitions between the hidden states can be explained as probabilities that the agent will either follow or change its current intention. Informally, this switching of intentions may be interpreted as follows: from an observer's perspective, sometimes suggesting that the observed agent is switching intentions appears to provide a better explanation of the behaviors.

We construct two HMM models: a normal model $\bar{N}$ and a suspicious model $\bar{S}$. We split all the labeled traces $\mathbf{x} \in D_l$ to traces generated by normal and suspicious agents, and use them to learn the parameters of the models $\bar{N}$ and $\bar{S}$, respectively. The model parameters can be locally optimized using an iterative procedure such as Baum-Welch method [13]. Given a new event trace $\mathbf{x}^{(k)} = (x_1, x_2, ..., x_k)$ we compute the probability that the trace was generated by each model $Pr\{\mathbf{x}^{(x)}|\bar{N}\}$ and $Pr\{\mathbf{x}^{(x)}|\bar{S}\}$ using a forward-backward procedure [13]. Given the prior probability $\bar{\lambda}$ we compute an estimate the trace $\mathbf{x}^{(k)}$ was generated by the suspicious process S:

$$Pr\{y = 1|H(t) = x_t, t = 1, ..., k\} = \frac{\bar{\lambda} \cdot Pr\{\mathbf{x}^{(k)}|\bar{S}\}}{\bar{\lambda} \cdot Pr\{\mathbf{x}^{(k)}|\bar{S}\} + (1-\bar{\lambda}) \cdot Pr\{\mathbf{x}^{(k)}|\bar{N}\}}. \tag{8}$$

Although the information about previous behavior is now partially encoded in the transition probabilities (i.e., given the agent's intention at time step t is suspicious it is more likely that the intention at $t+1$ will be suspicious as well), the model still uses the Markov assumption, i.e., the next agent's intention depends only on it's current intention. It is possible to introduce more complex HMM structures with long-term dependencies, but learning and inference in such models become computationally intractable [11].

5.3 Utility-Based Plan Recognition

We exploit UPR, an *Utility-based Plan Recognition*, briefly described below. The reader is referred to [3] for details. UPR consists of a plan library, which encodes behaviors of the observed agents in a form of directed graph, and a matching algorithm. It follows the footsteps of the hierarchical HMM in representing probabilistic information in the plan library. A plan step can be atomic, or non-atomic, i.e., broken down into atomic sub-steps, each a plan step in itself. Plan steps are linked via sequential edges, describing the execution order of a given plan and its sub-steps. UPR introduces three types of utilities on the edges: (a) the sequential utility from the current step to the next; (b) the interruption utility from the current step to the end of the plan; and (c) the decomposition utility from the current step at current level to its first substep at the sub-level. A corresponding probability is maintained for each type of utility. The observation sequence o is matched against the library using a *Symbolic Plan Recognizer* [2], which filters hypotheses that are consistent with o. Finally, the hypotheses are ranked by their expected utility.

We use a heuristic version of UPR as follows. Let $\hat{s}(x_t) = 1 - \hat{n}(x_t)$ be the probability that the trigger event x_t was generated by a suspicious person. Let $c_s > 0$ be the cost of the damage caused by a suspicious person if we do not stop him, and similarly, let $d_n = 0$ be the cost of the damage caused by a normal person. The expected cost of letting this person go (marking him as normal) is $c_{go} = c_s\hat{s}(x_t) + d_n\hat{n}(x_t) = c_s\hat{s}(x_t)$. Now suppose $c_n > 0$ is the cost of arresting an innocent person and $d_s = 0$ is the cost of the damage caused by a suspicious person when arrested. The expected cost of stopping this person (marking him as suspicious) is $c_{stop} = c_n\hat{n}(x_t) + d_s\hat{s}(x_t) = c_n\hat{n}(x_t)$. If there was only one event, we would compare both hypotheses and choose the one with the lowest expected cost. Supposing in this case $c_n\hat{n}(x_t)$ is lower, we would call this person suspicious.

One possible approach, based on the above expected-cost calculation, would be to determine whether a trigger event is to be categorized as suspicious or normal, and then to accumulate the total number of suspicious events, and subtract the total number of normal events; unfortunately, this simple strategy performs poorly. Therefore, not only do we count whether an event is suspicious or normal, but we give it a weight, proportional to the benefit or cost accrued. The function U_{UPR} hence evaluates an event trace $\mathbf{x}^{(k)}$ of a person by accumulating the weighted benefit of stopping this person and subtracting the weighted cost of arresting a normal person:

$$U_{UPR}(\mathbf{x}^{(k)}) = \sum_{t=1}^{k} b(x_t), \tag{9}$$

$$b(x_t) = \begin{cases} c_s\hat{s}(x_t); & \text{if } c_n\hat{n}(x_t) \leq c_s\hat{s}(x_t) \\ -c_n\hat{n}(x_t); & \text{if } c_n\hat{n}(x_t) > c_s\hat{s}(x_t) \end{cases}. \tag{10}$$

If the accumulated cost exceeds a threshold value τ', the person (i.e., trace $\mathbf{x}^{(k)}$) is marked as suspicious.

This remains a heuristic approach and further investigations could be a topic for future work; however, given that our next approach performs significantly superior, we chose to investigate that in more detail rather than providing more heuristics for the current approach.

5.3.1 Utilities as Potential Functions

Although the evaluation function U_{UPR} is well behaved, the utilities are constant and hence do not allow a dynamic adjustment to the behavior of the agent in the past. Thus, for instance, the first time we note a suspicious event, and the second time we note the same agent making a suspicious event, count equally. These utilities, however, are unable to express the characteristics of the empirical observations. Therefore, we extend the notion of utility and define the utility U as follows.

Definition 7. The utility function U over a plan step q_a, a plan step q_b, and the entire observation sequence $\mathbf{x}^{(t)}$ until current time step t is a function

$$U : \langle q_a, q_b, \mathbf{x}^{(t)} \rangle^n \to \mathbb{R}.$$

Utility function can be written as

$$U(q_a, q_b, \mathbf{x}^{(t)}) = \sum_{j=1}^{n} \lambda_j u_j(q_a, q_b, \mathbf{x}^{(t)}),$$

where each utility function u_j can be sequential, interruption, decomposition or any other utility, and λ_j are parameters to be defined. This allows us to introduce a set of auxiliary utility functions u_j describing not only the plan-step transitions but also the additional characteristics of the observation sequence. For example, the sequential utility from step q_i to q_{i+1} can be written as $u_t(q_i, q_{i+1}, \mathbf{x}^{(t)}) = c$, but in general, the constant c can be replaced with any function over q_i, q_{i+1} and $\mathbf{x}^{(t)}$.

Lemma 1. U is a well behaved function iff

$$\forall u_j, j = 1...k : u_j \text{ is well a behaved function.}$$

PROOF. Consider two well behaved functions f and g, and two scalar constants λ_f and λ_g. Let $f' = \lambda_f f$. Since multiplication with scalar preserves well-behaved property, f' is also a well behaved function. Let function u denote $u = f' + g'$. Then, $u(\mathbf{x}^{(t)}, x_{t+1}) = f'(\mathbf{x}^{(t)}, x_{t+1}) + g'(\mathbf{x}^{(t)}, x_{t+1}) \geq u(\mathbf{x}^{(t)}) = f'(\mathbf{x}^{(t)}) + g'(\mathbf{x}^{(t)})$ if $\Delta(x_{t+1} = 1)$, since f and g are well behaved and therefore $f'(\mathbf{x}^{(t)}, x_{t+1})$ and $g'(\mathbf{x}^{(t)}), x_{t+1})$ are non-negative. Similarly, f' and g' are non-positive when $\Delta(x_{t+1}) = 0$. □

5.3.2 Observation Utility for Suspicious Behavior Detection

In order to include the past behavior of an agent in an evaluation of the evidence, the utility function must be defined over the observation sequence. We propose an observation utility function that assigns cost using the number of normal and suspicious events in the past. Consider the example from Section 4. Suppose we see a person do a full U-turn in front of a police officer and we give this event a cost of 1. Later we see the same person doing a half-turn in front of a police officer. This event if seen on its own, would be given cost 0.5. However, following this initial turn where we had given a cost of 1, this new turn, becomes a 1 instead of 0.5. So, a linear accumulation would have given us a cost of 1.5, whereas because we bias the new event to register higher on our scale, our cost is 2 instead of 1.5.

Let $\eta_s(\mathbf{x}^{(k)})$ define the number of suspicious events in an event trace $\mathbf{x}^{(k)}$:

$$\eta_s(\mathbf{x}^{(k)}) = \sum_{t=1}^{k} \Delta(x_t), \tag{11}$$

Similarly, let $\eta_n(\mathbf{x}^{(k)}) = k - \eta_s(\mathbf{x}^{(k)})$ represent the number of normal events. Suppose we observed a trace $\mathbf{x}^{(k)}$ of all the suspicious events, i.e., $\forall t, t = 1, ..., k : \Delta(x_t) = 1$. Intuitively, the likelihood that an event x_t was indeed generated by a suspicious process increases exponentially according to the number of suspicious events in the past. On the other hand, if the events in $\mathbf{x}$ were normal, i.e., $\forall t, t = 1, ..., k : \Delta(x_t) = 0$, the likelihood exponentially decreases as the number of normal events increases. We define an observation utility function u_o over the current event x_t and trace $\mathbf{x}^{(t-1)}$ recursively as follows:

$$\begin{aligned} u_o(x_t, \mathbf{x}^{(t-1)}) &= \psi(\mathbf{x}^{(t)}) \cdot (u_o(\mathbf{x}^{(t-1)}) + \omega(\mathbf{x}^{(t)})), \quad (12) \\ u_o(\mathbf{x}^{(0)}) &= 0, \\ \omega(\mathbf{x}^{(t)}) &= \alpha \cdot \eta_s(\mathbf{x}^{(t)})^{s(x_t)/\beta}, \quad (13) \\ \psi(\mathbf{x}^{(t)}) &= \gamma \cdot \rho^{-\eta_n^*(\mathbf{x}^{(t)})/\eta_s(\mathbf{x}^{(t)})}. \quad (14) \end{aligned}$$

The term $\omega(\mathbf{x}^{(t)})$ uses an exponential function to assign a cost to the likelihood $s(x_t)$ that an event is suspicious. The parameter $\alpha > 0$ is the initial cost, η_s corresponds to the growth factor, and the parameter $0 < \beta < 1$ is the likelihood required for the cost to increase by the growth factor. The parameters α and β are estimated from the data. Suppose we observe two full U-turns, the second U-turn attributes higher cost to the overall suspicion, since the exponent base is increased due to the first U-turn.

Additionally, the term $\psi(\mathbf{x}^{(t)})$ employs an exponential time decay function that discounts the accumulated cost at time t according to the number of consecutive normal events η_n^*. The modified η_n^* represents *the time elapsed* since the last event $\Delta(x_i) = 1$, i.e., the number of normal events since the last suspicious event. The higher the number of consecutive normal events, the faster the cost decay. The parameter $0 < \gamma \leq 1$ is the initial decay, the parameter $0 < \rho < 1$ is the decay factor, and η_s is used to specify the number of events required for the decay to decrease by the decay factor. The parameters γ and ρ are also estimated from the data. Suppose we observe two agents, one already having made two U-turns and the other with only one U-turn. Suppose we observe both agents do a clearly normal event. The overall suspicion of the first agent is reduced less than the overall suspicion of the second agent. Hence the higher the number of suspicious events, the slower the suspicion decay.

The function u_o is a well-behaved function by definition. Eq. (12) can be rewritten, which gives us the utility function U_{F-UPR}:

$$\begin{aligned} U_{F-UPR}(\mathbf{x}^{(k)}) &= \sum_{t=1}^{k} \sum_{j=1}^{n} \lambda_j f_j(\mathbf{x}^{(t)}, q(t-i), q(t)) \\ &= \sum_{t=1}^{k} (\omega(\mathbf{x}^{(t)}) \prod_{i=t}^{k} \psi(\mathbf{x}^{(i)})). \quad (15) \end{aligned}$$

6. EXPERIMENTAL EVALUATION

We conducted empirical tests in a simulated airport domain to evaluate the performance of suspicious-passenger detection generated by four candidate algorithms. In addition, we compared the best two algorithms on the dangerous-driver domain [2].

To run proof-of-concept tests we considered a simulated environment, mainly to avoid difficulties due to privacy and confidentiality issues, and as well as due to the absence of real-world annotated data of suspicious behavior. A simulator also made it possible to control the amount of noise otherwise introduced by various vision systems (occlusions, false detections, etc.), and provided controllable and repeatable situations.

6.1 Airport domain

The experiments in this paper use the ESCAPES [15], a state-of-the-art, multiagent simulator for airport evacuations with several types of agents exhibiting behaviors of regular travelers, authorities, and families. The agents' behavior incorporates emotional, informational and behavioral interactions, such as emotional contagion, the spread of knowledge/fear, social comparison, etc. Therefore, an agent is affected by the behavior of other agents and their emotional states, and faced with uncertainty as to what happened and where the nearest exits are. We assume that the behavior of the agents corresponds to the behavior of real passengers at the airport.

In cooperation with security officials we defined a scenario where a suspicious passenger goes from point A to point B while trying to avoid security personnel at the airport. One may argue that an adversary that plans to do something malicious would behave normally in the presence of authorities, and this might be true for a highly trained individual. As discussed previously, an average person exposed to a high level of stress produces behavior that indicates fear, anxiety, tension, etc., and hence tries to cover it by minimizing close-range interactions by making u-turns, avoidance maneuvers, hiding in nearby shops, etc. Implementation details are provided on a supplemental web page[1].

A simulation in ESCAPES is run with a given airport map, authority agents, regular passengers and a suspicious agent going from point A to B, outputting traces with 2D coordinates for all agents. We initialized the simulator with 100 agents including $K_a \in \{5, 10, 15, 20, 25\}$ authorities and a suspicious person with randomly chosen initial and final points. For each K_a setting we ran 30 simulations, each consisting of $1500 - 3000$ time steps and 100 traces. On average, there were 215 interactions between the authorities and the passengers per run. To avoid issues that arise with highly unbalanced datasets we used random re-sampling without replacement to balance the data to the ratio *suspicious : normal* $= 20 : 80$.

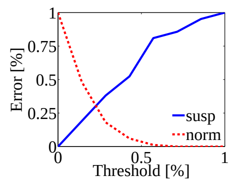
The trace of the coordinates was preprocessed to the action trace as follows. A change in position from the previous to the current state was described as taking the action of moving North, South, East and West, and their combinations (nine in total). This transformation describes the shape of a trajectory but discards the location information, which leads to better generalization. We also experimented with other transformations, for example, a more general one that also discards the orientation (forward, backward, left, right), and a less general one that divides the airport map with a square-based grid with numbered squares [2]. Preliminary tests showed the best performance when using the first transformation.

For the evaluation we used *precision*, *recall*, *specificity* and *F-measure*. Precision is defined as the number of true positives (all suspicious cases correctly classified as suspicious) divided by the number of all cases marked as suspicious (true and false positives): $pr = TP/(TP+FP)$. A perfect score 1 means that all cases marked as suspicious were indeed suspicious. Hence, the score $1 - pr$ represents the rate of *false alarms*. Recall is defined as the number of true positives divided by the number of all the suspicious cases: $re = TP/(TP+FN)$. A perfect score 1 means that all the suspicious cases were detected (but says nothing about falsely marked normal cases). Similarly, the specificity is defined for normal cases $sp = TN/(TN+FP)$. There are two points of interest, depending on our objective. The first one is when both scores are minimized, i.e., the trade-off point between false alarms and non-detected suspicious passengers, which can be detected with the F-measure $FM = 2 \cdot pr \cdot re/(pr+re)$. The other case is when a high false-alarm rate is acceptable and non-detected cases are extremely costly. In this case we are interested in precision when recall $re = 1$, i.e., all the suspicious passengers are found. In the worst-case scenario, all the passengers are marked as suspicious. We evaluate the statistical significance of our results using the two-sample t-test.

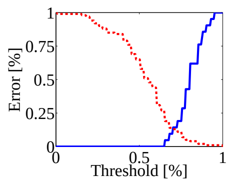
6.1.1 *Results*

In the first experiment we fixed the number of authority figures

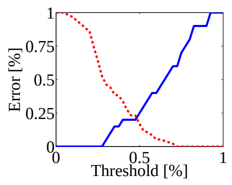
[1]http://dis.ijs.si/bostjan/aamas2012



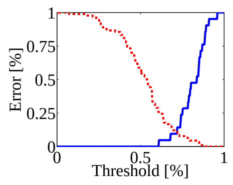
(a) $\exists k$ rule



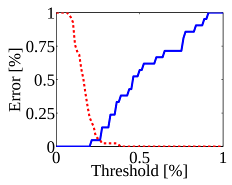
(b) Naive Bayes



(c) HMMs



(d) UPR



(e) F-UPR

Figure 3: Confusion error rates for different threshold values.

$K_a = 10$. We instantiated the naive Bayes, HMMs, UPR, and F-UPR detectors. Additionally, we considered another baseline detector using a simple rule over the threshold k and the event trace $\mathbf{x}^{(t)}$, saying that if the number of suspicious events exceeds k (i.e., $\exists k : \eta_s(\mathbf{x}^{(t)}) > k$), then mark trace $\mathbf{x}^{(t)}$ as suspicious. All the detectors used the event-trace probabilities $s'(\mathbf{x}^{(t)})$ and $n' = 1 - s'(\mathbf{x}^{(t)})$ as returned by the event-detection step. For the HMM approach we considered two ergodic HMMs as described is Section 6.1.2. We used two observations, the normal $\Delta(x_t) = 0$ and the suspicious $\Delta(x_t) = 1$ event, and varied the number of hidden states. The best results were achieved with three hidden states. Note that the HMMs detector applied on top of the CHMMs detector basically presents a version of the mixed layered HMM structure. All the models (including UPR and F-UPR detectors) were evaluated with 10-fold-cross validation.

Figures 3(a)–3(e) show the confusion error rates for suspicious (1-recall) and normal (1-specificity) passengers as a function of the normalized threshold value for all the five algorithms. For example, if the threshold is zero, then all the passengers are marked as suspicious. In this case: (i) all the suspicious passengers are correctly identified as suspicious, hence the error rate is also zero; and (ii) all the normal passengers are incorrectly identified as suspicious, hence the error rate is 1. As the threshold value increases, the error rate for correctly identifying the suspicious passengers increases, while the error rate for correctly identifying the normal passengers decreases.

There are two points of interest: (i) when the error rates cross each other, i.e., the F-measure is maximized; and (ii) the rightmost point when the error rate for suspicious passengers is zero (i.e., $re = 1$) and the other one is minimized. These cases are tabulated in Table 1. The first case is summarized in columns 2-4 showing the recall, precision and F-measure. F-UPR outperforms the $\exists k$ rule ($p < 0.01$), naive Bayes ($p < 0.01$), HMMs ($p < 0.01$), and UPR ($p < 0.01$). The second case, where the threshold value is such that all the suspicious passengers are discovered, is shown in columns 5-6. Column five shows the confusion error for normal passengers (i.e., 1-specificity), while the column six shows the ratio of correctly raised alarms (i.e., precision). The $\exists k$ rule, for instance, marks all the passengers as suspicious (FP rate is 100%) and consequentially almost 80% of alarms are false. HMMs achieve better performance, but still mark more than 50% of normal passengers as suspicious. Other methods mark between 1/5 and 1/4 of normal passengers as suspicious, but precision is around 50%, which means that every second passenger marked as suspicious is indeed suspicious (and all suspicious passengers are discovered!). Overall, F-UPR in this setting also outperforms the $\exists k$ rule ($p < 0.01$), naive Bayes ($p < 0.05$), HMMs ($p < 0.01$), and UPR ($p < 0.05$). Finally, Figure 4 depicts the ROC curves showing that F-UPR performs the same or better in all the threshold settings.

In the last experiment we varied the number of authorities in the simulation. We expect that an increased number of authority

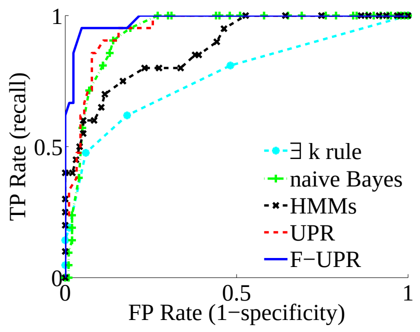
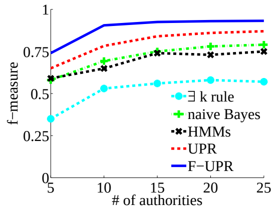


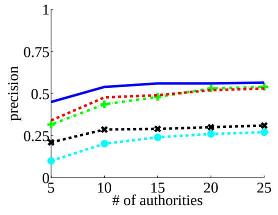
Figure 4: ROC curves comparing all the detectors.

Table 1: Evaluation results when the F-measure is maximized (columns 2-4) and all the suspicious cases are discovered (last two columns).

	max FM			re=1	
Algorithm	re	pr	FM	1-spec	pr
$\exists k$ rule	0.619	0.464	0.530	1.000	0.202
Naive Bayes	0.857	0.581	0.693	0.270	0.436
HMMs	0.600	0.706	0.649	0.526	0.286
UPR	0.857	0.720	0.783	0.256	0.477
F-UPR	0.905	0.905	0.905	0.217	0.539



(a) F-measure is maximized.



(b) All suspicious passengers are discovered.

Figure 5: Evaluation results for varying the number of authority figures in the simulation and two different threshold values.

figures will result in more interactions between the suspicious passengers and the authorities, which will make detection easier. Figure 5 shows the results for the $K_a \in \{5, 10, 15, 20, 25\}$ authority figures in a simulation: Fig. 5(a) shows the F-measure for a threshold such that the F-measure is maximized, while Fig. 5(b) shows the precision when $re = 1$. An increased number of authority figures first significantly increases the detection capabilities. For example, the F-measure for F-UPR increases by 15% when the security resources are doubled from five to ten, but as the number increases, the impact is smaller. We can also see that F-UPR achieves the same performance as other methods using significantly less security resources.

6.1.2 Detection Based on the Action Trace

We also applied a sanity check and tested the suspicious behavior detection from a sequence of agent's actions (i.e., action trace **a**) instead of a sequence of trigger events (i.e., event trace **x**). We used HMMs, since they are considered as a baseline for modeling a sequence of actions. The goal is to differentiate between a sequence of actions produced by a suspicious and a regular passenger. We expect this approach not to perform well, since it is too general and unable to precisely model the interactive behavior present in a multiagent environment.

The suspicious behavior detector consists of two ergodic HMMs: S' trained on the suspicious and N' trained on the regular action traces. A new trace is first transformed to the action trace $\mathbf{a}^{(k)}$ as described previously and then matched against both HMMs, yielding the likelihood that it produced the given $\mathbf{a}^{(k)}$. If the likelihood is greater than a threshold the action trace is marked as suspicious. We tested this approach for $K_a = 10$. At the threshold value s.t. the highest F-measure of 18.01 was achieved this approach achieved an acceptable discovery rate ($re = 66.23$) and an extremely low precision ($pr = 10.42$). Such a performance positions this approach under the $\exists k$ rule. The overall performance was consistent with our expectations. Modeling single-agent actions in a multiagent environment is not able to capture the interactive behavior.

6.2 Catching a Dangerous Driver

In addition to the airport domain we applied UPR and F-UPR to the dangerous-driver domain, as introduced in [2]. This domain also includes behavior that becomes increasingly costly if repeated; a driver switching a lane once or twice is not necessarily acting suspiciously, but a driver zigzagging across two lanes is dangerous. Our goal was to detect such drivers as soon as possible.

We generated 100 observation sequences (each of N observations) of a zigzagging driver, and 1000 sequences of a safe driver. The observations were sampled with 10% noise from the trajectories. If the driver stayed on the same lane as in the previous sample, the event was considered as normal, otherwise it was considered as dangerous. For each sequence of trigger events we accumulated the associated cost using both UPR and F-UPR.

Table 2 reports the performance at the peak F-measure for different lengths of the observation sequence. The results confirm the experiments on the airport domain for two points. First, F-UPR performs better than UPR for any selected sequence length. Second, the performance of both methods increases as the number of observations increases, where F-UPR requires fewer observations than UPR to achieve the same performance.

7. CONCLUSION

This paper successfully addressed the problem of suspicious behavior detection from a set of observations, where no single observation suffices to make the decision. The paper addresses the problem in two steps, i.e., the detection of trigger events and a combination of evidence to reach the final decision. To that end, the main contributions of this paper are: (i) the conditions that a reasonable detector should satisfy; (ii) an analysis of three detectors; (iii) a novel F-UPR approach that extends the notion of utilities; and (iv) comprehensive experiments on two simulated domains. By providing a new algorithm that outperforms other approaches, this paper has advanced the state of the art.

Table 2: Performance at the peak F-measure in dangerous driver domain.

Sequence length N	F-UPR	UPR
25	0.632	0.540
50	0.720	0.667
75	0.900	0.800
100	0.952	0.857
125	1.000	0.947

8. REFERENCES

[1] D. Arsić, B. Schuller, and G. Rigoll. Suspicious behavior detection in public transport by fusion of low-level video descriptors. In *Proc. of the 8th ICME*, pages 218–221, 2007.

[2] D. Avrahami-Zilberbrand. *Efficient Hybrid Algorithms for Plan Recognition and Detection of Suspicious and Anomalous Behavior*. PhD thesis, Bar-Ilan University, 2009.

[3] D. Avrahami-Zilberbrand and G. A. Kaminka. Incorporating observer biases in keyhole plan recognition (efficiently!). In *AAAI*, 2007.

[4] M. Brand, N. Oliver, and A. Pentland. Coupled hidden Markov models for complex action recognition. In *CVPR*, pages 994 – 999, 1997.

[5] T. V. Duong, H. H. Bui, D. Q. Phung, and S. Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-Markov model. In *CVPR*, pages 838–845, 2005.

[6] F. Esponda, S. Forrest, and P. Helman. A formal framework for positive and negative detection schemes. *IEEE Systems Man and Cybernetics Society*, 34(1):357–373, 2004.

[7] R. S. Feris, A. Hampapur, Y. Zhai, R. Bobbitt, L. Brown, D. A. Vaquero, Y. li Tian, H. Liu, and M.-T. Sun. *IBM Smart Surveillance System*. Taylor & Francis Group, 2009.

[8] C. W. Geib and R. P. Goldman. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, 173(11):1101–1132, 2009.

[9] P. Helman and G. Liepins. Statistical foundations of audit trail analysis for the detection of computer misuse. *IEEE Transactions on Software Engineering*, 19(9):886–901, 1993.

[10] S. Hongeng and R. Nevatia. Large-scale event detection using semi-hidden Markov models. In *IEEE Int. Conf. on Computer Vision*, pages 1455–1462, Aug. 2003.

[11] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[12] M. Naylor and C. I. Attwood. Advisor: Annotated digital video for intelligent surveillance and optimised retrieval, 2003. Final report.

[13] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *IEEE*, 1989.

[14] G. Sukthankar and K. Sycara. Hypothesis pruning and ranking for large plan recognition problems. In *AAAI*, 2008.

[15] J. Tsai, G. Kaminka, S. Epstein, A. Zilka, I. Rika, X. Wang, A. Ogden, M. Brown, N. Fridman, M. Taylor, E. Bowring, S. Marsella, M. Tambe, and A. Sheel. ESCAPES - Evacuation simulation with children, authorities, parents, emotions, and social comparison. In *AAMAS*, 2011.

[16] N. Vaswani, A. R. Chowdhury, and R. Chellappa. Shape activity: A continuous state HMM for moving/deforming shapes with application to abnormal activity detection. In *IEEE Trans. on Image Processing*, pages 1603 – 1616, 2005.

Session 1F
Planning

Probabilistic Planning with Non-Linear Utility Functions and Worst-Case Guarantees [*]

Stefano Ermon, Carla Gomes, Bart Selman
Department of Computer Science
Cornell University
{ermonste,gomes,selman}@cs.cornell.edu

Alexander Vladimirsky
Department of Mathematics
Cornell University
vlad@math.cornell.edu

ABSTRACT

Markov Decision Processes are one of the most widely used frameworks to formulate probabilistic planning problems. Since planners are often risk-sensitive in high-stake situations, non-linear utility functions are often introduced to describe their preferences among all possible outcomes. Alternatively, risk-sensitive decision makers often require their plans to satisfy certain worst-case guarantees.

We show how to combine these two approaches by considering problems where we maximize the expected utility of the total reward subject to worst-case constraints. We generalize several existing results on the structure of optimal policies to the constrained case, both for finite and infinite horizon problems. We provide a Dynamic Programming algorithm to compute the optimal policy, and we introduce an admissible heuristic to effectively prune the search space. Finally, we use a stochastic shortest path problem on large real-world road networks to demonstrate the practical applicability of our method.

Categories and Subject Descriptors

I.2 [**ARTIFICIAL INTELLIGENCE**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms, Theory

Keywords

Planning, Utility Functions, Constraints

1. INTRODUCTION

Markov Decision Processes (MDPs) are one of the most widely used frameworks to formulate probabilistic planning problems. In these problems, the notion of risk is related to the fact that, given the stochastic nature of the problem, each policy can generally produce several possible outcomes, and some of them might reflect unsatisfactory performance. In many applications, such as space planning and natural resource management, it is critical to use performance metrics that allow the ability to manage the risk, i.e. a certain level of control over unfavorable outcomes [4, 18].

The problem of managing the risk has been studied extensively in artificial intelligence, operations research, and control theory. Many formulations have been proposed (see Section 2 for more details), among which decision theoretic planning and worst-case approaches are the two most widely used. The former is based on decision theory, more specifically, on the fact that decision makers accepting a small number of axioms always choose the course of actions that maximizes the expected utility of the total reward [16], where the specific form of the utility function describes the risk attitude of the planners. The latter is focused on providing deterministic guarantees for the plans by looking at worst-case realizations of the random processes involved.

In this paper, we show how to combine these two approaches by considering problems where the objective is to maximize the expected utility of the total reward subject to worst-case, linear constraints. For example, in the case of a linear utility function, we can maximize the expected total reward only among those policies whose reward is larger than a given threshold, even in the worst-case scenario. With a (non-linear) "step" utility function, we can maximize the probability of reaching a target reward level, while enforcing the worst-case constraint at the same time.

Our theoretical results extend previous work on MDPs with non-linear utility functions and show that the optimal policy for the constrained optimization problem is highly structured: it is deterministic, and even though generally not Markovian, it depends on the history only through the total accumulated reward. Therefore, an optimal policy can be represented (and approximated) much more effectively than general history-dependent policies. Furthermore, we show how to exploit the presence of worst-case constraints to define an admissible heuristic, which we use in a Dynamic Programming algorithm to speed up the policy search.

To demonstrate the practical applicability of our method, we consider stochastic shortest path problems as a special case of MDPs. We show that our algorithm scales to large real-world road networks, and it leads to plans that are significantly different from the ones obtained with traditional optimization criteria. We think this type of formulation can be particularly useful for time-dependent problems, such as the ones faced by the Green Driver App [2], where traffic light information are explicitly modeled. In fact, in this situation it is necessary to consider policies that are non-Markovian, even when given linear utility functions.

[*]Supported by NSF Grants 0832782 and DMS-1016150.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

2. RELATED WORK

Decision theoretic planning has been studied extensively, mainly in artificial intelligence [18, 14], control theory, and operations research [10, 15, 1, 7]. Typically, monotonically non-decreasing utility functions, mapping total rewards to utility values, are used to to describe the preferences of the planner. In particular, exponential utility functions [10] are commonly used because they satisfy a separability property that allows an efficient Dynamic Programming solution. Recently, researchers have also considered planning problems with more general non-linear utility functions [12]. The results in this paper are related to that line of work (and we use a similar notation whenever possible), but with the novel introduction of worst-case constraints.

The most conservative approach to account for risk is worst case planning, where only the worst possible outcome is optimized. A generalization known as α-value criterion is introduced in [9], where outcomes that happen with a probability smaller than α are not considered (when $\alpha = 0$, it is equivalent to the worst case). In this work, instead of optimizing the worst-case scenario, we introduce constraints that need to be satisfied by the plan, under all possible realizations of the randomness. The relationship of our approach with worst-case planning is discussed in detail below in Section 4.1.

There are several existing frameworks for constrained probabilistic planning problems in the literature. Many of them [1, 7] involve the maximization of an expected (discounted) reward subject to upper bounds on the total expected (discounted) costs. The main limitation of this approach is that upper bounding an expected value might provide a guarantee in terms of risk that is too weak, because it constitutes only a mild restriction on the possible outcomes (constraints are satisfied only on average, while our constraints are met by all possible realizations). The same holds for mean-variance analysis [17], where the problem is analyzed in terms of the tradeoff between expected value and variance of the total reward (either by imposing constraints on the variance, or associating a cost with it). The constrained formulation that is closest to our work is the sample-path constraint introduced in [15]. In [15], they consider time-average MDPs, with a reward and cost associated with each decision. The optimization problem is to maximize the expected average reward over all policies that meet the sample-path constraint, where a policy is said to meet the sample-path constraint if the time-average cost is below a specified threshold with probability one. Notice that also in this case the guarantee can be quite weak, because the constraint is imposed only on an averaged quantity. Finally, in [8] they derive and solve Dynamic Programming equations for two special types of worst-case constrained Stochastic Shortest path problems (in our formalism these correspond to U_L and $U_{K,L}$ utility functions defined in section 6). We emphasize that the new framework in this paper is significantly more general and can be applied to general MDPs to provide worst-case performance guarantees with general non-linear utility functions.

3. PROBLEM DEFINITION

We consider probabilistic planning problems represented as Markov Decision Processes. Formally, an MDP is a tuple (S, A, P, r) where S is a set of states, A is a set of actions, P is a set of transition probabilities and $r : S \times A \times S \mapsto \mathbb{R}$ is an (immediate) reward function. If an agent executes an action $a \in A$ while in a state $s \in S$, then it receives an immediate reward $r(s, a, s')$ and it transitions to a new state $s' \in S$ with probability $P(s'|s, a)$. We denote by $A_s \subseteq A$ the set of actions available while the agent is in state s.

In this paper we consider *finite* MDP where both the state space S and action space A are finite sets.

Policies. Let the planning horizon T be the (possibly infinite) number of time steps that the agent plans for. A *history* at time step t is a sequence $h_t = (s_0, a_0, \cdots, s_{t-1}, a_{t-1}, s_t)$ of states and actions that leads from the initial state s_0 to state s_t at time step t. The set of all histories at time step t is denoted $H_t = (S \times A)^t \times S$.

In a probabilistic setting, a plan is represented by a *policy*, where a policy is a sequence of decision rules, one for each time step in the planning horizon. The most general decision rules are *randomized history-dependent* (HR), which are mappings $d_t : H_t \to P(A)$, where $P(A)$ is the set of probability distributions over the set of actions A. A history-dependent decision rule is called *Markovian* if it depends only on the current state s_t, while it is called *deterministic* if it deterministically choses the action to be taken. A policy is called *stationary* if $d_t = d$ for all time steps t within the planning horizon and d is a Markovian decision rule. We denote the class of *deterministic stationary* (SD) policies by Π^{SD} and the class of *randomized stationary* policies by Π^{SR}.

Utility Functions. Let w_T be the total reward received by the agent, that is the sum of all the immediate rewards accrued within the planning horizon

$$w_T = \sum_{t=0}^{T-1} r_t(s_t, a_t, s_{t+1}) \tag{1}$$

A standard approach to model the preferences of the planner among the possible realizations of w_T is to use a monotonically non-decreasing utility function $U : \mathbb{R} \mapsto \mathbb{R}$, which maps total rewards to utility values. Decision theory suggests that decision makers accepting a small number of axioms always choose the course of actions that maximizes the expected utility of the total reward [16].

4. FINITE HORIZON PROBLEMS

First we consider planning problems where the planning horizon T is finite, and we will later extend the results to the infinite horizon case. We define the *value* of a policy $\pi \in \Pi^{HR}$ from an initial state $s \in S$ as

$$v^{\pi}_{U,T}(s) = \mathbb{E}^{s,\pi}\left[U\left(\sum_{t=0}^{T} r_t\right)\right] = \mathbb{E}^{s,\pi}\left[U\left(w_T\right)\right] \tag{2}$$

which is the expected utility of the total reward w_T. For standard utility functions, the expected utilities exist and are finite because there is a finite number of possible finite trajectories in finite MDPs [14]. The optimal values

$$v^{*}_{U,T}(s) = \sup_{\pi \in \Pi} v^{\pi}_{U,T}(s) \tag{3}$$

exist since the *values* exist for every policy $\pi \in \Pi$. Properties of the optimal policy for this case have been studied in [12]. In particular, the optimal policy is deterministic and even though it is generally not Markovian, it depends on the history h_t only through the accumulated reward w_t.

4.1 Worst-Case Constraints

Risk-sensitive planners are often interested in worst-case scenarios. With this perspective, a common approach is to look for a policy that maximizes the worst case performance (*game against nature*). Formally, in the max-min version of the problem, we seek to optimize

$$d_T^*(s) = \sup_{\pi \in \Pi} d_T^\pi(s) \tag{4}$$

where

$$d_T^\pi(s) = \min \{k | \mathbb{P}[w_T = k] > 0\} \tag{5}$$

is the worst-case realization of the total reward w_T (the definition is well posed since w_T is a discrete random variable with a finite sample space for a finite MDP).

In this paper, we consider situations where the planner wants to enforce linear worst case constraints on the total reward w_T of the form

$$w_T > L \tag{6}$$

for all possible realizations of w_T (equivalently, (6) has to hold almost surely, because w_T has a finite sample space). Notice that the *game against nature* approach is equivalent to finding the largest value of L such that condition (6) can be met.

In this paper we combine the problem of finding a policy that maximizes the expected utility, with the presence of linear worst-case constraints on the total reward. Formally, we wish to find

$$v_{U,T,L}^*(s) = \sup_{\pi \in \Pi(L)} v_{U,T}^\pi(s) \tag{7}$$

where the optimization is restricted to the set of policies $\Pi(L)$ whose corresponding total reward w_T satisfies condition (6), for all possible realizations of w_T. The problem is well defined when the set $\Pi(L)$ is not empty, that is if and only if $L < d_T^*(s)$.

As an example, in the simplest case of a linear utility function $U(x) = x$, the objective is to maximize the total expected reward but only among those policies with a guaranteed lower bound L on the total reward.

4.2 Extended-Value Utility Functions

We show that the constrained problem defined by Equation (7) can be solved by considering the original optimization problem defined by Equation (3) with a more general utility function. We introduce the concept of an *extended-value utility function*, which can be used to model linear worst case constraints on the total reward w_T. Let $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty\}$ be the affinely extended real number system, which turns into a totally ordered set by defining $-\infty \leq a \leq +\infty$ for all $a \in \mathbb{R}$. We define an *extended-value utility function* a *monotonically nondecreasing* function $U : \mathbb{R} \to \overline{\mathbb{R}}$ that maps wealth levels to the corresponding utility values.

Let us consider the problem previously defined by Equations (2) and (3) in the more general case where U is an *extended-value utility function.* Recall that $\mathbb{E}[X]$ exists and $\mathbb{E}[X] = \mathbb{E}[X_+] - \mathbb{E}[X_-]$ when $\mathbb{E}[X_+] < \infty$ or $\mathbb{E}[X_-] < \infty$, where X_+ and X_- denote the positive and negative part of X, respectively. In the more general case of an *extended-value utility function*, the expected utilities defined by (2) exist (but are not always finite), because $\mathbb{E}[U(w_T)_+] < \infty$ since the number of trajectories is finite for finite MDPs.

Notice that agents acting as though they were maximizing expected *extended utility functions* satisfy the *Completeness* and *Transitivity* axioms of the Von Neumann-Morgenstern Utility Theorem [16], because the extended real number system is totally ordered and the order relation is transitive. Moreover, they satisfy the *Indipendence* axiom by the linearity of expectation. Notice however that they violate the *Continuity* axiom. In fact, given three *lotteries* such that $A \succeq B \succeq C$ (A is preferred over B, and B over C) and where the expected utilities of A and B are finite but the expected utility of C is $-\infty$, there is no combination of A and C that gives an expected utility that is equal to the one of B.

The optimal values defined by (3) need not to be finite, but they are bounded from above since the total rewards are bounded and thus the expected utilities of the total reward for all policies are bounded from above as well. The following result holds:

LEMMA 1. *For any extended value utility function U, let $L = \sup\{w | U(w) = -\infty\}$. Then for any policy $\pi \in \Pi^{HR}$, $v_{U,T}^\pi(s) > -\infty$ if and only if $w_T > L$ almost surely.*

PROOF. If $P[w_T \leq L] > 0$, it follows $v_{U,T}^\pi(s) = -\infty$. If $w_T > L$ almost surely, then it follows by monotonicity that $v_{U,T}^\pi(s) > U(L) \geq -\infty$. □

As a corollary, if the optimal value is finite, then the worst case constraint $w_T > L$ is satisfied by the optimal plan.

Using Lemma 1, we show that we can solve the constrained problem defined by Equation (7) for a standard utility function U by solving the unconstrained problem (3) with an *extended-value* utility function U_e defined as follows

$$U_e(x) = \begin{cases} -\infty & x \leq L \\ U(x) & x > L \end{cases}$$

LEMMA 2. *For any utility function U and lower bound L such that $\Pi(L)$ is non empty, $v_{U,T,L}^*(s) = v_{U_e,T}^*(s)$.*

PROOF. Let π' be the optimal policy for the constrained problem. Since $\pi' \in \Pi(L)$, we have $w_T > L$ almost surely and therefore

$$-\infty < v_{U,T,L}^*(s) = v_{U,T,L}^{\pi'}(s) = v_{U_e,T}^{\pi'}(s) \leq v_{U_e,T}^*(s)$$

Since $v_{U_e,T}^*(s) = v_{U_e,T}^{\pi^*}(s) > \infty$, by Lemma 1, π^* satisfies the constraint (6) almost surely, so $\pi^* \in \Pi(L)$ and

$$v_{U,T,L}^*(s) \geq v_{U_e,T}^{\pi^*}(s) = v_{U_e,T}^*(s)$$

□

We focus now on characterizing the optimal policy for the unconstrained problem with an extended value utility function U_e (we drop the subscript for compactness). In the rest of the paper, we will use subscripts to indicate the length of the planning horizon T and the utility function U used. We will use superscripts to indicate the policy π used and, when relevant, the decision epoch t the value refers to.

4.3 Optimality Conditions

We provide a characterization of the optimal policies for maximum expected utility planning problems by generalizing some results obtained in [12] to the more general case of extended value utility functions.

THEOREM 1. *Let $\pi = (d_0, \dots, d_{T-1}) \in \Pi^{HR}$ be a policy. The values $v^{\pi,t}_{U,T}(h_t) = \mathbb{E}^{\pi}\left[U(w_T)|h_t\right]$ of a policy π at time step t given an history $h_t \in H_t$ satisfy*

$$v^{\pi,T}_{U,T}(h_T) = U\left(w_T\right), h_T \in H_T$$

$$v^{\pi,t}_{U,T}(h_t) = \sum_{a\in A_{s_t}} d_t(h_t,a) \sum_{s'\in S} P(s'|s_t,a) v^{\pi,t+1}_{U,T}(h_t \circ (a,s'))$$

where $h_t \in H_t, 0 \le t < T$, $\circ$ is the composition operator, and the last component of h_t is s_t.

PROOF. Similar to Theorem 4.1 in [12]. □

If we define the optimal values for a history h_t as $v^{*,t}_{U,T}(h_t) = \sup_{\pi\in\Pi} v^{\pi,t}_{U,T}(h_t)$, then we have $v^{*,0}_{U,T}(s) = v^{*}_{U,T}(s)$ for any initial state s. Furthermore,

THEOREM 2. *The values $v^{*,t}_{U,T}(h_t)$ are the unique solutions to the optimality equations*

$$v^{*,T}_{U,T}(h_T) = U\left(w_T\right), h_T \in H_T$$

$$v^{*,t}_{U,T}(h_t) = \max_{a\in A_{s_t}} \sum_{s'\in S} P(s'|s_t,a) v^{*,t+1}_{U,T}(h_t \circ (a,s')) \qquad (8)$$

for $h_t \in H_t, 0 \le t < T$ and where the last component of h_t is s_t.

PROOF. Similar to Theorem 4.2 in [12]. □

The above results also show that there exists a *deterministic* history-dependent optimal policy, that for an history h_t chooses a maximizer in Eq. (8) as action. However, the policy might depend on the entire previous history h_t. In the following sections we use the state-augmentation approach to show that the policy has more structure, i.e. it depends on the history only through the total reward accumulated so far $w_t = \sum_{k=0}^{t-1} r_k(s_k, a_k, s_{k+1})$.

4.4 State Space Augmentation

A deeper characterization of the structure of the optimal policy can be obtained by considering a new *augmented* MDP where the state space is augmented with wealth levels (corresponding to the sum of accumulated rewards). As in [12], let

$$R = \{0\} \cup \{r(s,a,s')|P(s'|s,a) > 0, s, s' \in S, a \in A_s\}$$

be the set of possible rewards. Then the set W^t of all possible wealth levels at time step t is inductively defined as follows

$$W^0 = \{0\}, W^{t+1} = \{r + w | r \in R, w \in W_t\}$$

We consider an extended MDP where the augmented states space is $\langle S\rangle = (S\times W^0) \cup (S \times W^1) \cup \dots \cup (S\times W^T)$. The actions available in an augmented state $\langle s\rangle = (s,w)$ are $A_{\langle s\rangle} = A_{(s,w)} = A_s$ for all wealth levels w. The transition probability from a state $\langle s\rangle = (s,w)$ to $\langle s\rangle' = (s', w')$ is

$$P(\langle s\rangle' \mid \langle s\rangle, a) = \begin{cases} P(s'|s,a) & \text{if } w' = w + r(s,a,s') \\ 0 & \text{otherwise} \end{cases}$$

All augmented rewards $r(\langle s\rangle, a, \langle s\rangle')$ are zero, and there is a terminal augmented reward $J(\langle s\rangle) = U(w)$ applicable at time T for an augmented state $\langle s\rangle = (s,w)$. There is no utility function for the augmented model, so the value $\langle z\rangle^{\langle\pi\rangle}_T(s,w)$ of an augmented policy $\langle\pi\rangle$ from initial augmented state (s,w) is given by the expected total augmented reward (equivalently, by the expected augmented terminal reward, since all other augmented rewards are zero).

Notice that the construction previously used in [12] cannot be used in our generalized case because it defines rewards in the augmented model as the difference of two utilities, which might not be well defined in for an extended value utility function. Notice also that the augmented MDP is still finite for a finite planning horizon T.

The original MDP and the augmented one are closely related, and intuitively the two underlying stochastic processes are equivalent. Formally, it can be shown as done in [12] that there is a 1-1 mapping between a history of the original model and a class of equivalent histories of the augmented model.

LEMMA 3. *For any wealth level w, and for any history of the original model $h_t = (s_0, a_0, s_1, \cdots, s_t) \in H_t$, the sequence $\phi_w(h_t) = \langle h\rangle_t = (\langle s\rangle_0, a_0, \langle s\rangle_1, \cdots, \langle s\rangle_t)$ is a history of the augmented model, where*

$$\langle s\rangle_k = (s_k, \tilde{w}_k) = (s_k, w + w_k), 0 \le k \le t$$

Furthermore, for any history of the augmented model $\langle h\rangle_t = (\langle s\rangle_0, a_0, \langle s\rangle_1, \cdots, \langle s\rangle_t) \in \langle H\rangle_t$ where $\langle s\rangle_k = (s_k, \tilde{w}_k)$ for all $0 \le k \le t$, there exists a wealth level w such that $\tilde{w}_k = w + w_k$ and the sequence $\psi(\langle h\rangle_t) = (s_0, a_0, s_1, \cdots, s_t)$ is a history of the original model.

PROOF. Similar to Lemmas 4.3 and 4.4 in [12]. □

Similarly, using Lemma 3, for any policy in the original model $\pi = (d_0, d_1, \cdots, d_{T-1}) \in \Pi^{HR}$, we define a policy of the augmented model, $\Psi(\pi) = (\langle d\rangle_0, \langle d\rangle_1, \cdots, \langle d\rangle_{T-1})$, such that for all augmented histories $\langle h\rangle$, $\langle d\rangle_t(\langle h\rangle, a) = d_t(\psi(\langle h\rangle), a)$.

For any augmented policy $(\langle d\rangle_0, \langle d\rangle_1, \cdots, \langle d\rangle_{T-1})$, we define a policy in the original model, $\Phi_w = (d_0, d_1, \cdots, d_{T-1})$, such that for all histories $h \in H_t$,

$$d_t(h,a) = \langle d\rangle_t(\phi_w(h), a)$$

Furthermore, the values of the policies in the original and augmented model are closely related:

THEOREM 3. *For each policy $\pi \in \Pi^{HR}$ in the original MDP and for all states $s \in S$,*

$$\langle z\rangle^{\Psi(\pi)}_T(s,w) = \mathbb{E}^{s,\pi}\left[U(w + \sum_{t=0}^{T-1} r_t)\right]$$

For each policy $\langle\pi\rangle$ in the augmented MDP, for each wealth level $w \in W$,

$$\langle z\rangle^{\langle\pi\rangle}_T(s,w) = \mathbb{E}^{s,\Phi_w(\langle\pi\rangle)}\left[U(w + \sum_{t=0}^{T-1} r_t)\right]$$

PROOF. First, we need to prove the probabilistic equivalence of the stochastic processes induced by policies that correspond through the mappings Ψ and Φ_w. The proof is similar to the one of Theorem 4.5 in [12] and is omitted. Furthermore,

$$\langle z\rangle^{\Psi(\pi)}_T(s,w) = \mathbb{E}^{(s,w),\Psi(\pi)}\left[\sum_{t=0}^{T-1}\langle r\rangle_t + J(\widetilde{s}_T, \widetilde{w}_T)\right] =$$

$$\mathbb{E}^{(s,w),\Psi(\pi)}\left[U(\widetilde{w}_T)\right] = \mathbb{E}^{s,\pi}\left[U(w + \sum_{t=0}^{T-1} r_t)\right]$$

where $(\widetilde{s}_T, \widetilde{w}_T)$ is the terminal augmented state, because both policies produce equivalent random processes. For the second part,

$$\langle z\rangle_T^{\langle\pi\rangle}(s,w) = \mathbb{E}^{(s,w),\langle\pi\rangle}\left[\sum_{t=0}^{T-1}\langle r\rangle_t + J(\widetilde{s}_T,\widetilde{w}_T)\right] =$$

$$\mathbb{E}^{(s,w),\langle\pi\rangle}\left[J(\widetilde{s}_T,\widetilde{w}_T)\right] = \mathbb{E}^{s,\Phi_w(\langle\pi\rangle)}\left[U(w+\sum_{t=0}^{T-1} r_t)\right]$$

because of the probabilistic equivalence. □

Since the augmented MDP is a standard MDP, it is well known [3] that the optimal values $\langle z\rangle_T^*(s,w)$ exist (but are not necessarily finite) for all augmented states (s,w). Moreover, there exists a Markovian, deterministic policy $\langle\pi\rangle_T^*$ that is optimal for the augmented model. We show that $\Phi_0(\langle\pi\rangle_T^*)$ is an optimal policy for the original MDP (optimality for the original MDP refers to the maximum expected utility criterion):

$$\begin{aligned} v_{U,T}^*(s) &= v_{U,T}^{\pi_T^*}(s) = \langle z\rangle_T^{\Psi(\pi_T^*)}(s,0) \\ &\leq \langle z\rangle_T^*(s,0) = \langle z\rangle_T^{\langle\pi\rangle_T^*}(s,0) = v_{U,T}^{\Phi_0(\langle\pi\rangle_T^*)}(s) \end{aligned}$$

Notice that the optimal policy $\Phi_0(\langle\pi\rangle_T^*)$ for the original model is not Markovian anymore. However, the dependency on the history h_t is limited, since the decision rules d_t only depend on the accumulated reward w_t.

It is also very important for the optimal values to be finite. Policies that do not meet the worst-case requirements (i.e., with infinite values for the expected utility) all have the same value, even though they might not perform equally badly.

4.5 Policy Computation and Pruning

The augmented problem previously described is a standard Markov Decision Process, where the objective is to maximize the total expected reward. Therefore, the optimal policy $\langle\pi\rangle_T^*$ can be computed using Dynamic Programming equations, as shown in Algorithm 1.

Algorithm 1 Dynamic Programming equations for the augmented problem

```
$t \leftarrow T$
for all $s \in S$ do
  Initialize $\langle z\rangle_T^{*,T}(s,w) = U(w)$
for $t = T-1 \to 0$ do
  for all $s \in S$ do
    for all $w \in W^t$ do
      $\langle z\rangle_T^{*,t-1}(s,w) =$
      $\max_{a\in A_s}\sum_{s'\in S} P(s'|s,a)\left[\langle z\rangle_T^{*,t}(s', w + r(s,a,s'))\right]$
```

Notice also that $\langle z\rangle_T^{*,t}(s,w) = -\infty$ whenever

$$w + d_{T-t}^*(s) \leq L \tag{9}$$

where $d_k^*(s)$ are the optimal max-min values defined by Equation (4). Intuitively, it means that $\langle z\rangle_T^{*,t}(s,w) = -\infty$ whenever we cannot meet the worst-case requirement, not even when optimizing the worst-case performance. Using condition (9), we can introduce additional pruning in a Forward Dynamic Programming algorithm, where the optimal values $\langle z\rangle_T^{*,t}(s,w)$ are recursively computed according to

$$\langle z\rangle_T^{*,t}(s,w) = \max_{a\in A_s}\sum_{s'\in S} P(s'|s,a)\left[\langle z\rangle_T^{*,t+1}(s', w + r(s,a,s'))\right]$$

with the two base cases:

$$\langle z\rangle_T^{*,t}(s,w) = \begin{cases} U(w) & \text{if } t = T \\ -\infty & \text{if } w + d_{T-t}^*(s) \leq L \end{cases}$$

once we precomputed the optimal max-min values $d_k^*(s)$, for all $s \in S$ and $0 \leq k \leq T$.

5. INFINITE HORIZON

For an infinite horizon planning problem, the *value* of a policy $\pi \in \Pi^{HR}$ is defined as

$$v_U^\pi(s) = \lim_{T\to\infty} v_{U,T}^\pi(s) = \lim_{T\to\infty}\mathbb{E}^{s,\pi}\left[U\left(\sum_{t=0}^{T} r_t\right)\right] \tag{10}$$

In general, the limit is neither guaranteed to exists nor to be finite, even in the standard case of a real-valued utility function [13].

However, in the special case of *Negative MDPs* (where all rewards are non-positive, i.e. $r(s,a,s') \leq 0$), we can prove the existence of the limit in Equation (10) in the general case of an extended value utility function. In fact, we already proved that the expectation exists for each T, and the existence of the limit derives from the monotonicity of the utility function U and w_T.

As in the finite horizon case, it is crucial that the optimal values are finite, because otherwise plans cannot be compared in a meaningful way based on their expected utility. We therefore provide sufficient conditions that guarantee the finiteness of the optimal values.

We consider a special class of infinite horizon goal directed MDPs where there is a finite set of *goal states* $G \subseteq S$, where the agent stops to execute actions, and no longer receives rewards. Further, we restrict ourselves to the case of *negative MDPs*, where $r(s,a,s') \leq 0$.

5.1 Finiteness

Let's consider the infinite horizon version of the max-min problem previously described. Let

$$d^\pi(s) = \lim_{T\to\infty} d_T^\pi(s)$$

where $d_T^\pi(s)$ is defined according to Equation (5). Again, we can prove that the limit exists by monotonicity. Let $d^*(s) = \sup_\pi d^\pi(s)$ be the optimal worst-case value. By definition, under the optimal worst-case policy π^{WC} we have $d^*(s) \leq w_T$ for all $T \geq 0$ and initial states s, so for each $T \geq 0$

$$U(d^*(s)) \leq v_{U,T}^{\pi^{WC}}(s) \leq v_{U,T}^*(s) \leq U(0) \tag{11}$$

It follows that if $U(d^*(s))$ is finite, then from (11) we have that $v_U^*(s)$ exists and is also finite. Intuitively, this condition means that the worst-case constraint encoded with the extended value utility function cannot be too restrictive, that is it cannot be more restrictive than what is possible to achieve using the optimal worst-case policy.

5.2 Properties of the Optimal Policy

We consider a *negative goal-directed MDP* that satisfies condition (11), with the additional condition $r(s,a,s') < 0$ for all $s \in S \setminus G$ (*strictly negative rewards*). Let

$$L = \sup\{w | U(w) = -\infty\} > -\infty$$

We show there exists an optimal policy for the Maximum Expected Utility objective that is *stationary*, deterministic and that depends on the history h_t only through the accumulated reward w_t. Let

$$\overline{r} = \max_{s \in S\setminus G} \max_{a \in A_s} \max_{s' \in S} r(s,a,s') < 0, \; \overline{T} = \lceil L/|\overline{r}| \rceil + 1$$

The following result holds:

Lemma 4. *For any policy $\pi \in \Pi^{HR}$ for the infinite horizon problem ($T = \infty$) with strictly negative rewards, for any state $s \in S$, and for all $T' \geq \overline{T}$,*

$$v^{\pi}_{U,T'}(s) = v^{\pi}_{U,\overline{T}}(s)$$

Proof. Let w_T be defined as in (1). By monotonicity, $v^{\pi}_{U,T'}(s) \leq v^{\pi}_{U,\overline{T}}(s)$. If $v^{\pi}_{U,\overline{T}}(s) = -\infty$, we are done. Otherwise, it must be $w_{\overline{T}} \geq L$ almost surely. By the definition of $\overline{T}$ and L, it must be the case that $s_{\overline{T}} \in G$ almost surely. This concludes the proof because $r_k(s_k, a_k, s_{k+1}) = 0$ almost surely for any $k \geq \overline{T}$, because we must have reached a goal state. $\square$

Using Lemma 4 and taking limits, we have that for any policy $\pi \in \Pi^{HR}$ and for all initial states, $\lim_{T' \to \infty} v^{\pi}_{U,T'}(s) = v^{\pi}_{U,\overline{T}}(s)$. Therefore,

$$\sup_{\pi \in \Pi} \lim_{T' \to \infty} v^{\pi}_{U,T'}(s) = \sup_{\pi \in \Pi} v^{\pi}_{U,\overline{T}}(s) = v^{*}_{U,\overline{T}}(s)$$

which means that we can solve the infinite horizon problem by planning for a finite horizon of length $\overline{T}$, for instance using Algorithm 1. Using the results in Section 4 and Lemma 4, the optimal policy for the infinite horizon problem is deterministic and history-dependent (but the dependency on the history is only through the accumulated reward). Furthermore, we show it is stationary.

Lemma 5. *For any state $s \in S$, and wealth level w in the augmented MDP problem we have*

$$\langle z \rangle^{*,t_1}_{\overline{T}}(s,w) = \langle z \rangle^{*,t_2}_{\overline{T}}(s,w)$$

Proof. Let $\langle \pi \rangle^{*}_{\overline{T}} = (\langle d_0 \rangle, \cdots, \langle d_{\overline{T}-1} \rangle)$ be the optimal policy for the augmented problem. By the optimality principle, $\langle z \rangle^{*,t}_{\overline{T}}(s',w') = \langle z \rangle^{*}_{\overline{T}-t}(s',w')$. Without loss of generality, let $t_1 < t_2 \leq \overline{T}$. By previous theorems, monotonicity of U and negative rewards assumption

$$\langle z \rangle^{*}_{\overline{T}-t_1}(s,w) = \mathbb{E}^{\langle s \rangle, \langle \pi \rangle^{*}_{\overline{T}-t_1}}[U(w + \sum_{k=0}^{T-1-t_1} r_k)] \leq$$

$$\mathbb{E}^{\langle s \rangle, \langle \pi \rangle^{*}_{\overline{T}-t_1}}[U(w + \sum_{k=0}^{T-1-t_2} r_k)] \leq \langle z \rangle^{*}_{\overline{T}-t_2}(s,w)$$

If $\langle z \rangle^{*,t_2}_{\overline{T}}(s,w) = -\infty$ then we are done. Otherwise, $\tilde{w}_{\overline{T}-t_2} = w + \sum_{k=0}^{\overline{T}-1-t_2} r_k \geq L$ almost surely when using policy $\langle \pi \rangle^{*}_{\overline{T}-t_2}$ from the initial state (s,w). If $s \in G$ we are done because $\langle z \rangle^{*,t_1}_{\overline{T}}(s,w) = \langle z \rangle^{*,t_2}_{\overline{T}}(s,w) = U(w)$. Otherwise if s is not a goal state and since $w \in W^{t_2}$, then it must be $w \leq t_2 * \overline{r}$. If $\tilde{x}_k$ is the state after k steps when using policy $\langle \pi \rangle^{*}_{\overline{T}-t_2}$ from the initial state (s,w), it must be $\tilde{x}_{\overline{T}-t_2} \in G$ almost surely, because otherwise the reward $\tilde{w}_{\overline{T}-t_2}$ would exceed the bound. Then the following policy $\langle \pi \rangle'_{\overline{T}-t_1} = (\langle d_{\overline{T}-t_2} \rangle, \cdots, \langle d_{\overline{T}-1} \rangle, \cdots)$ satisfies

$$\langle z \rangle^{*}_{\overline{T}-t_1}(s,w) \geq \langle z \rangle^{\langle \pi \rangle'_{\overline{T}-t_1}}_{\overline{T}-t_1}(s,w) = \langle z \rangle^{*,t_2}_{\overline{T}}(s,w)$$

because $\tilde{x}_{\overline{T}-t_2} \in G$ almost surely. $\square$

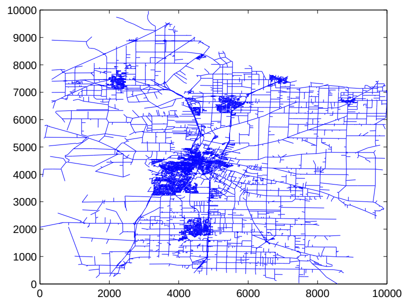


Figure 1: San Joaquin County Road Network

As a corollary, the optimal policy for the augmented MDP is *stationary* and therefore the optimal policy for the original problem is also *stationary*. Intuitively, since we are given an infinite number of steps to reach the goal, the number of steps already taken does not affect the optimal policy.

Notice that under these assumptions (i.e., with a worst case constraints), we can compute the optimal policy for the augmented problem using Algorithm 1, which terminates in a bounded number of steps. In contrast, using the construction in [12], one has to reach a fixed-point using value-iteration procedures in a (countably) infinite state space, which in general requires some form of approximation.

6. STOCHASTIC SHORTEST PATHS IN ROAD NETWORKS

In a Stochastic Shortest Path problem, a planner agent is given a graph with vertex set V and edge set E, an initial node $s \in V$, and a set of goal nodes $G \subseteq V$. From a node $s \in V \setminus G$, the agent can move to any neighboring node (this is the set of available actions), but unlike standard shortest path problems, the cost of traversing an edge $e \in E$ is stochastic and modeled by a random variable c_e with known probability distribution. The planner stops when a goal node $g \in G$ is reached, and no more costs are incurred. Given an utility function U (see examples below), the goal of the agent is to find a plan that maximizes the expected utility of the total reward, which is defined as minus the total cost. Notice that the problem can be formulated as a finite MDP when the random variables $\{c_e, e \in E\}$ are discrete with finite sample space.

We consider a real-world road network [11] as the underlying graph (see Figure 1 for an example) in our experiments. The edge lengths $\{w_e, e \in E\}$ are also provided in the dataset. The edge costs $\{c_e\}$ model travel times, and are assumed to be discretized Beta-distributed random variables. This is a common modeling assumption for tasks with unknown duration in PERT analysis [6]. In particular, we assume $c_e = m + (M - m)\mathbf{B}(\alpha, \beta)$, where $\mathbf{B}$ follows a Beta distribution with shape parameters α and β. M and m are respectively the upper and lower bound on c_e, and are defined as follows:

$$M = (1 + u_1(e)/2)\, w_e$$
$$m = (1 - u_2(e)/2)\, w_e$$

where $u_1(e)$ and $u_2(e)$ are uniformly distributed in $[0, 1]$. The parameters α and β are chosen such that the expected edge cost is equal to the edge length for each $e \in E$ (i.e. $E[c_e] = w_e$), with a variance chosen uniformly at random. Note that the rewards $\{r_e\}$ are a *discretized* version of $\{-c_e\}$, so that the finiteness MDP assumption holds.

6.1 Utility Functions for SSPs

In our experiments, we consider several types of utility functions. A simple linear utility function $U(x) = x$ leads to the standard maximization of the expected total reward. To maximize the expected total reward (equivalently, minimize expected travel time) with a worst case constraint we use

$$U_L(x) = \begin{cases} -\infty & x \le L \\ x & x > L \end{cases} \tag{12}$$

In the *Stochastic On Time Arrival* formulations [5], also known as MDPs with Target-Level Utility Functions [12], the utility function has the form $U^K(x) = 1_{[K,+\infty)}(x)$ where $1_{\mathcal{A}}$ is an indicator function for the set $\mathcal{A}$. This corresponds to maximizing the probability of reaching a certain target reward K, since it holds that $\mathbb{E}[U^K(w_T)] = \mathbb{E}[1_{[K,+\infty)}(w_T)] = \mathbb{P}[w_T \ge K]$ To maximize the probability of having a total reward at least as large as K with a guaranteed lower bound $L < K$, we can introduce an extended value utility function $U_{K,L}(x)$ that is defined to be $-\infty$ when $x \le L$, and equal to $U^K(x)$ otherwise. When costs represent travel times, this corresponds to maximizing the probability of reaching the destination by a given deadline, with a worst case constraint. For instance, we might wish to use this criterion in order to maximize the probability of getting to the airport at least 3 hours before our flight departure, but no later than check-in closure time. We can also consider a more general case where the deadline is soft (as in [12]), because partial credit is given for being late, up to some point D. We introduce a worst-case constraint L by using the following *extended-value* utility function:

$$U_{K,D,L}(x) = \begin{cases} 1 & K \le x \\ (x - D)/(K - D) & D \le x < K \\ 0 & L < x < D \\ -\infty & x \le L \end{cases}$$

Finally, we consider a worst-case constrained exponential utility function $U_{\gamma,L}(x)$, by introducing a worst-case lower bound L in the standard utility function $U_\gamma(x) = e^{\gamma x}$.

6.2 Results

For our experiments we use the San Joaquin County Road Network graph (with 18263 nodes and 23874 edges) represented in Figure 1. Every policy $\pi \in \pi^{HR}$ has an associated probability distribution for the total reward w_T. Assuming the costs $\{c_E\}$ represent travel times, this corresponds to a probability distribution for the total travel time $c_T = -w_T$.

For each utility function previously introduced, we compute the corresponding optimal policy with a worst case constraint L using the forward Dynamic Programming method. The optimal max-min values $d_k^*(s)$ are precomputed solving a shortest path problem on the original graph with edge costs given by the worst-case realization. Given a fixed initial position $s \in V$ and destination node $G = \{g\}$, each optimal policy has a different associated probability distribution for the total travel time c_T (notice that they are all optimal, but according to different criteria). In Figure 2, we compare the resulting probability distributions (obtained optimizing different performance metrics), and we also emphasize their worst-case realization (the dashed vertical line on the right). For comparison, we also provide the probability distribution corresponding to the optimal worst-case policy π^{WC} (in red). For Markovian policies (such as π^{WC}), the probability distribution is computed exactly (by evaluating a convolution), while distributions associated with history-dependent policies are obtained by Monte Carlo sampling with 100,000 samples (in green).

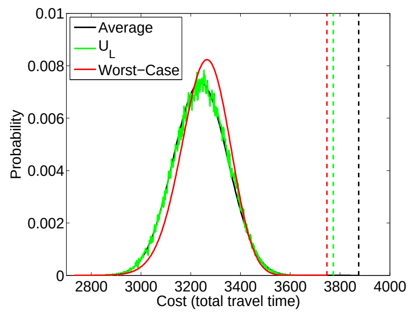
First, we compare the standard linear utility function $U(x) = x$ with its worst-case constrained version $U_L(x)$ defined as in Equation (12). In Figure 2a we see the results for a source-destination pair s, g where we improve the worst-case realization of w_T, while at the same time maintaining the same expected value $\mathbb{E}[w_T]$. In other words, the policy for $U_L(x)$ dominates the one for $U(x) = x$ because it achieves the same expected value but it improves the worst-case performance. However, it is not always the case. In Figure 2b, we see that for a different source-destination pair, improving the worst-case realization of w_T leads to a larger expected travel time $\mathbb{E}[c_T] = \mathbb{E}[-w_T]$.

Finally, in Figure 2c and 2d we plot the probability distributions corresponding to $U_{K,D,L}(x)$ (maximizing the probability of reaching the destination by a given soft deadline with a worst-case constraint L) and $U_{\gamma,L}(x)$. In both cases, the distributions are significantly different from the one obtained minimizing the expected travel time (in black). Notice that in Figure 2c the probability of reaching the destination by the deadline is significantly improved, and that there is a spike around $c_T = 3800$. This is because according to $U_{K,D,L}(x)$, any realization of c_T larger than $-D$ has the same utility, as long as they satisfy the worst-case requirement. Similarly, the exponential utility function $U_{\gamma,L}(x)$ reflects a strong preference for small realizations of c_T. This can be seen in Figure 2d, where for instance the probability $P[c_T < 2900]$ of having a total travel time smaller than 2900 is 4 times larger when optimizing $U_{\gamma,L}(x)$ rather than $U(x) = x$ (area under the green and black curve, respectively). This experiment empirically demonstrates that when the planner cares about different objectives (e.g., a target level criterion), then augmented policies can achieve significant improvements over standard Markovian ones.

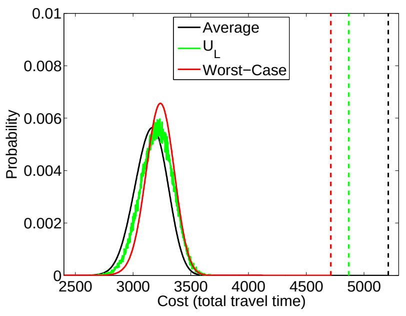
7. CONCLUSIONS

In this paper, we combined aspects of the two most widely used frameworks to model risk-sensitive planning, i.e. maximum expected utility and worst-case (*games against nature*) formulations. We introduced a new class of problems, where the goal is to maximize the expected utility of the total reward w_T, subject to a linear worst-case constraint on w_T. We showed how to encode this constraint using an *extended value utility function* in a maximum expected utility formulation, and we proved several results on the structure of the corresponding optimal policy.

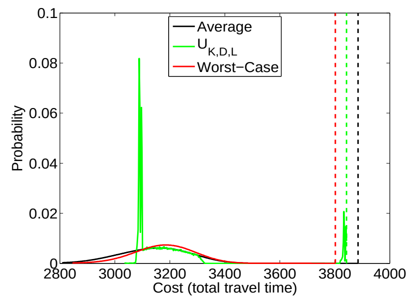
We showed that for finite planning horizons and for a class



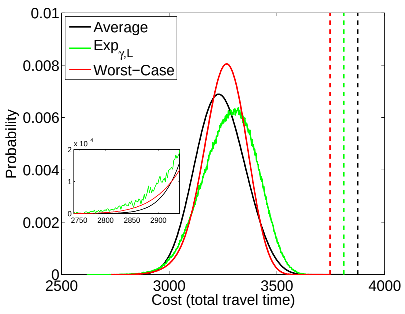
(a) Constrained linear, $L = -3773$



(b) Constrained linear, $L = -4867$



(c) Constrained soft deadline, $D = -3300,\ K = -3100,\ L = -3843$



(d) Constrained exponential, $\gamma = 0.02, L = -3811$

Figure 2: Resulting probability distributions and worst-case bounds (dashed lines). See pdf for colored version.

of infinite horizon problems, the optimal policy is deterministic and although not Markovian, it depends on the history only through the accumulated reward. Therefore, the policy can be represented as a set of functions (one for each state $s \in S$) of the total reward w, which, if necessary, can be approximated much more effectively than general history dependent decision rules, i.e. functions defined on the set of all possible histories H_t.

Although introducing non-linear utility functions allows the expression of a richer set of planning preferences, it increases the complexity because of the augmentation of the state space. However, adding worst-case constraints does not further increase the complexity, and allows us to speed up the policy search algorithm with additional pruning.

We think this type of formulation can be particularly useful for time-dependent problems where using the augmented space is unavoidable. For instance, in the Green-Driver App [2], they face SSPs where the edge costs probability distributions are dependent on the current time (essentially on w_t) because they model traffic lights. Although we used synthetic edge cost probability distributions, we showed our approach scales to large real-world networks, and it leads to significantly different plans with respect to traditional optimization criteria.

8. REFERENCES

[1] E. Altman. *Constrained Markov decision processes.* Chapman & Hall, 1999.

[2] J. Apple, P. Chang, A. Clauson, H. Dixon, H. Fakhoury, M. Ginsberg, E. Keenan, A. Leighton, K. Scavezze, and B. Smith. Green driver: AI in a microcosm. In *AAAI*, 2011.

[3] D. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, 2005.

[4] S. Ermon, J. Conrad, C. Gomes, and B. Selman. Playing games against nature: optimal policies for renewable resource allocation. *UAI*, 2010.

[5] Y. Fan, R. Kalaba, and J. Moore. Arriving on time. *J. Optimization Theory and Applications*, 127(3).

[6] N. Farnum and L. Stanton. Some results concerning the estimation of beta distribution parameters in PERT. *J. Operational Research Society*, 38(3):pp. 287–290, 1987.

[7] E. Feinberg and A. Shwartz. Constrained discounted dynamic programming. *Mathematics of Operations Research*, 21(4):922–945, 1996.

[8] M. Guay, T. Pham, D. Plotkin, S. Ermon, and A. Vladimirsky. Safer optimal routing in stochastic networks. *Unpublished technical report*, 2010.

[9] M. Heger. Consideration of risk in reinforcement learning. In *ICML*, volume 105, page 111, 1994.

[10] R. Howard and J. Matheson. Risk-sensitive Markov decision processes. *Management Science*, 18(7):356–369, 1972.

[11] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S. Teng. On trip planning queries in spatial databases. *Advances in Spatial and Temporal Databases.*

[12] Y. Liu. *Decision-theoretic planning under risk-sensitive planning objectives.* PhD thesis, Georgia Institute of Technology Atlanta, GA, USA, 2005.

[13] Y. Liu. Existence and finiteness conditions for risk-sensitive planning: Results and conjectures. In *UAI*, 2005.

[14] Y. Liu and S. Koenig. Probabilistic planning with nonlinear utility functions. In *ICAPS*, pages 410–413, 2006.

[15] K. Ross and R. Varadarajan. Markov decision processes with sample path constraints: the communicating case. *Operations Research*, pages 780–790, 1989.

[16] J. Von Neumann, O. Morgenstern, A. Rubinstein, and H. Kuhn. *Theory of games and economic behavior.* Princeton Univ Pr, 2007.

[17] D. White. Mean, variance, and probabilistic criteria in finite Markov decision processes: a review. *J. Optimization Theory and Applications*, 56(1):1–29, 1988.

[18] S. Zilberstein, R. Washington, D. Bernstein, and A. Mouaddib. Decision-theoretic control of planetary rovers. *Adv. in Plan-Based Control of Robotic Agents.*

Heuristic Search of Multiagent Influence Space

Stefan J. Witwicki
GAIPS / INESC-ID
Instituto Superior Técnico
Porto Salvo, Portugal
stefan.witwicki@ist.utl.pt

Frans A. Oliehoek
CSAIL, MIT / DKE, Maastricht University
Maastricht, The Netherlands
fao@csail.mit.edu

Leslie P. Kaelbling
CSAIL
MIT
Cambridge, MA 02139, USA
lpk@csail.mit.edu

ABSTRACT

Multiagent planning under uncertainty has seen important progress in recent years. Two techniques, in particular, have substantially advanced efficiency and scalability of planning. Multiagent heuristic search gains traction by pruning large portions of the joint policy space deemed suboptimal by heuristic bounds. Alternatively, influence-based abstraction reformulates the search space of joint policies into a smaller space of influences, which represent the probabilistic effects that agents' policies may exert on one another. These techniques have been used independently, but never together, to solve larger problems (for Dec-POMDPs and subclasses) than previously possible. In this paper, we take the logical albeit nontrivial next step of combining multiagent A* search and influence-based abstraction into a single algorithm. The mathematical foundation that we provide, such as partially-specified influence evaluation and admissible heuristic definition, enables an investigation into whether the two techniques bring complementary gains. Our empirical results indicate that A* can provide significant computational savings on top of those already afforded by influence-space search, thereby bringing a significant contribution to the field of multiagent planning under uncertainty.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

General Terms

Algorithms, Theory, Performance

Keywords

Multiagent Planning Under Uncertainty, Heuristic Search, Multiagent A*, Influence-Based Abstraction, TD-POMDP.

1. INTRODUCTION

Computing good policies for agents that are part of a team is an important topic in multiagent systems. This task, planning, is especially challenging under uncertainty, e.g., when actions may have unintended effects and each agent in the team may have a different view of the global state of the environment due to its private observations. In recent years, researchers have proposed to gain grip on the problem by abstracting away from *policies* of other agents and instead reasoning about the effects, or *influences*, of those policies [1, 2, 22, 23, 25]. However, no methods have been proposed to effectively search the space of influences other than enumeration. In this paper, we fill this void by showing how it is possible to perform heuristic search of the influence space, thereby significantly speeding up influence-based planning.

The problem of multiagent planning under uncertainty can be formalized as a decentralized partially observable Markov decision process (Dec-POMDP) [3]. However, its solution is provably intractable (NEXP-complete). As such, many methods either focus on finding approximate solutions without quality guarantees [10, 5, 13, 18, 22, 23], or providing optimal solutions for restricted subclasses. In particular, more efficient procedures have been developed for problems that exhibit *transition and observation independence* [2, 11, 11, 21] or *reward independence* [1]. Unfortunately, these subclasses are too restrictive for many interesting tasks, such as mobile agents collaborating in the search for a target.

The *transition-decoupled* POMDP (TD-POMDP) [25] has recently been introduced as a model that allows for transition, observation, and reward dependence, while still allowing for more efficient solutions than the general Dec-POMDP model. The core idea is to exploit independence between agents by formalizing the *influence* they can exert on each other. This abstract representation of interaction-related behavior parameterizes a search space of joint influences, which is often significantly smaller than the joint policy space (cf. [24] chapter 4) and, in principle, cheaper to search. Nevertheless, like the policy space, the influence space can still grow exponentially in problem size.

The challenge that we address here is how to search the influence space efficiently. Whereas previous TD-POMDP solutions have focused on exhaustive influence-space search, in general Dec-POMDPs, A* policy-space search guided by heuristics, i.e., multiagent A* (MAA*), has been shown to be an extremely powerful method for reducing the computation required to find optimal solutions [14, 19, 20]. The main contribution of this paper is to show how the strengths of heuristic policy-space search can be transferred to influence-space search.

To accomplish this, we make the following auxiliary contributions: we show how one can define heuristics in influence space, we prove the admissibility of such heuristics, thus guaranteeing optimality of A* search, and we provide the results of an empirical evaluation that shows that our proposed methods can yield significant performance increases,

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

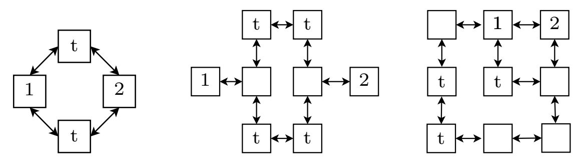


Figure 1: HOUSESEARCH environments. '1'/'2' marks search robot start positions. 't' marks possible static target locations.

	NMF	MMF
locally affected	x_i^l	m_i^l
nonlocally affected	N/A	m_i^n
unaffectable	x_i^u	m_i^u

Table 1: Different types of state factors that make up s_i.

especially on problems that are hard for exhaustive influence search. Additionally, we demonstrate how TD-POMDPs can be used for an important class of problems: locating objects or targets with a team of agents, which also leads us to the first application of influence search on problems that have cyclic dependencies between the agents.

2. INFLUENCE-BASED ABSTRACTION

After describing a motivating problem domain, we review the TD-POMDP model and influence-based policy abstraction, and explain how they can be exploited to find optimal solutions via optimal influence space search.

2.1 Motivating Domain: Locating Targets

Although the TD-POMDP model and the methods presented in this paper extend to other settings, in this paper we focus on their application to problems where a team of agents has to locate a target given a prior probability distribution over its location and a model of its movement. We assume that the target either remains stationary or moves in a manner that does not depend on the strategy used by the searching agents.

More concretely, we consider a problem domain called HOUSESEARCH in which a team of robots must find a target in a house with multiple rooms. Such an environment can be represented by a graph, as illustrated in Fig. 1. At every time-step, each agent i can stay in its current node n or move to a neighboring node n'. The location of agent i is denoted l_i and that of the target is denoted l_{target}. The movements, or actions a_i, of each agent i have a specific cost $c_i(l_i,a_i)$ (e.g., the energy consumed by navigating to a next room) and can fail; we allow for stochastic transitions $p(l_i'|l_i,a_i)$. Also, each robot receives a penalty c_{time} for every time step that the target is not found. When a robot is in the same node n as the target, there is a probability of detecting the target $p(detect_i|l_{target},l_i)$, an event which will be modeled by a state variable 'target found by agent i' (denoted f_i). When the target is detected, the agents receive a reward r_{detect}. Given the prior distribution and model of target behavior, the goal is to optimize the expected sum of rewards, thus trading off movement cost and probability of detecting the target as soon as possible.

2.2 TD-POMDP Model

Here we formalize the planning task for scenarios such as the HOUSESEARCH task described above. First, we introduce the single-agent factored POMDP, and then we describe how a TD-POMDP extends this model to multiple agents.

A *factored partially observable Markov decision process* for a single agent (indexed i for consistency with multiagent notation later on) is a tuple $\langle \mathcal{S}_i, \mathcal{A}_i, T_i, R_i, \mathcal{O}_i, O_i \rangle$, where $\mathcal{S}_i = X_1 \times \cdots \times X_k$ is the set of states s_i induced by a set of k state variables or *factors*, $\mathcal{A}_i$ is the set of actions that the agent can take, T_i is the transition model that specifies $\Pr(s_i'|s_i,a_i)$, $R_i(s_i,a_i,s_i')$ is the reward function, $\mathcal{O}_i$ is the set of observations o_i, and O_i is the observation function that specifies $\Pr(o_i|a_i,s_i')$. Because the state space is factored, it is usually possible to specify T_i, R_i and O_i in a compact manner using a Bayesian network called a two-stage temporal Bayesian network (2TBN) [4]. Given this model, the planning task for a POMDP is to find an optimal policy π that maximizes the expected sum of rewards over h time steps or stages. Such a policy maps from *beliefs*, probability distributions over states, to actions. While solving a POMDP is widely considered to be an intractable problem, in the last two decades many exact and approximate solution methods have been proposed (see, e.g., [7]).

Intuitively, a TD-POMDP is a *set* of factored POMDPs, one for each agent, where there is overlap in the state factors of each agent.[1] Moreover, the set of state factors can be divided into factors that occur only in one agent's local state space ('non-mutual' factors (NMFs)) and factors that are 'mutually modeled' by more than one agent (MMFs). A TD-POMDP imposes the restriction that each state factor can be *directly* affected by the action of at most one agent. That is, in the 2TBN, each factor can have an incoming edge from only 1 action variable. This does not mean that state factors depend on just one agent, since factors can be indirectly (i.e., via a directed path consisting of multiple edges) influenced by many agents. This leads to different parts of an agent's local state, as summarized in Table 1. Using the notation defined in this table, we will write the local state of an agent i as $s_i = \langle x_i^l, x_i^u, m_i^l, m_i^n, m_i^u \rangle = \langle x_i, m_i \rangle$. The joint reward function for the TD-POMDP is the summation of the individual reward functions for each agent's POMDP: $R(s,a) = \sum_i R_i(s_i,a_i)$, and we assume an initial joint state distribution $\boldsymbol{b}^0$. For a more formal introduction of the TD-POMDP framework, see [24, 25].

The 2TBN representation of the TD-POMDP's transition, observation, and reward model for HOUSESEARCH is illustrated in Fig. 2. Here, two search agent's local states overlap such that both model the target location and the factors f_1,f_2. Note that the mutually modeled state factors can only be characterized as (non)locally affected from the perspective of a particular agent. E.g., f_1 is locally affected for agent 1, but non-locally affected for agent 2. The figure also shows that, in this domain, each agent's reward function R_i can also be factored as the sum of two components R_{detect} and R_{move}. The former models the rewards for de-

[1]Of course, for such a setting to be well-defined means that different transition models T_i need to be consistent since the local transition probabilities can depend on other agents too. One can alternatively consider the existence of a single transition model T defined over the joint action and the joint state.

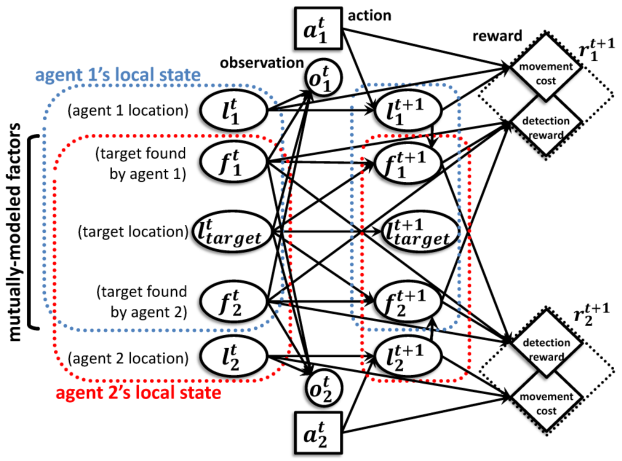


Figure 2: A TD-POMDP for HouseSearch.

tection, as well as the time cost (c_{time}) of not detecting. This component depends on f_1^{t+1}, f_2^{t+1} as well as on f_1^t, f_2^t: only when (at least) one of the f_i variables switches from false to true do the agents get the reward, when all four factors are false the agents get the time penalty and otherwise the rewards are 0 (but the movement costs remain). The movement reward components only depend on the agents' non-mutual locations and local actions.

The TD-POMDP is a non-trival subclass of the factored Dec-POMDP [15], for which the NEXP-completeness result still holds [24]. This also means that single-agent POMDP solution methods do not directly apply. Intuitively, in a multiagent context, we are now searching for a joint policy $\pi = \langle \pi_1, \ldots, \pi_n \rangle$. Moreover, an agent can no longer base its policy on a simple belief over states, as this does not account for the beliefs and actions of its teammates.

2.3 Influences and Local Models

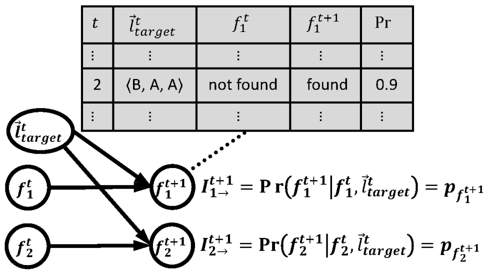
A well-known solution method for Dec-POMDPs, called JESP [10], searches for a locally optimal joint policy as follows: it starts with a random joint policy and then selects one agent to improve its policy while keeping the other policies fixed. The improvement of the selected agent is done by computing a best response via dynamic programming. From the perspective of a single agent i, by fixing π_{-i} (the policies of the other agents) the problem can be re-cast as an augmented POMDP, where the augmented state is a tuple $\langle s, \vec{o}_{-i} \rangle$ of a nominal state and the observation histories of the other agents.

Since a TD-POMDP is a Dec-POMDP, JESP directly applies. However, because of the special structure a TD-POMDP imposes, we can account for this structure to compute the best response in a potentially more efficient way: rather than maintaining a JESP belief $b_i(s, \vec{o}_{-i})$, agent i can maintain a condensed belief $b_i(s_i^t, \vec{m}_i^{t-1})$ over just its own local state and the history of mutually modeled factors [25]. Intuitively, this is possible, because all information about $\vec{o}_{-i}$ and the state factors that are not in agent i's local state (i.e., x_j for $j \neq i$) is captured by $\vec{m}_i^t$.[2] That is, $\vec{m}_i^t$ d-separates the agent's observation history $\vec{o}_i^t$ from those of other agents $\vec{o}_{-i}^{t-1}$. For instance, given the DBN connectivity in Fig. 2, all information agent 2 has about l_1^t is inferred from the history of f_1^t and l_{target}^t.

A second important observation is that an agent i is only influenced by other agents via its nonlocal mutually modeled factors m_i^n. E.g., in Fig. 2 agent 1 only influences agent 2 via the f_1 factor. Therefore, if, during planning, the value of this factor at all stages is known, agent 2 can completely forget about agent 1 and just solve its local POMDP (and similar for agent 1). This line of reasoning holds even if agent 2 does not know the exact values of f_1 ahead of time, but instead knows the probability that f_1 turns to true for each stage. This insight lies at the basis of *influence-based policy abstraction*: all policy profiles π_{-i} that lead to the same distributions over non-local MMFs $m_i^{n,0}, \ldots, m_i^{n,h-1}$ can be clustered together, since they will lead to the same best response of agent i.

To formalize this idea, an *incoming influence point* of agent i, denoted $I_{\rightarrow i}$, specifies a collection of conditional probability tables (CPTs): one for each nonlocally affected MMF, for each stage $t = 1, \ldots, h-1$.[3] We denote a CPT for f_1^t (from our example) as $p_{f_1^t}$, which specifies probabilities $p_{f_1^t}(v|\cdot)$ for values $v \in \{0,1\}$ of f_1^t given its parents $(\cdot)$. In this example, $I_{\rightarrow 2} = \{p_{f_1^1}, p_{f_1^2}, \ldots, p_{f_1^{h-1}}\}$. To specify these CPTs, it is only necessary to use $\vec{m}_i$, the history of mutual features, as the parents [25]. I.e., the CPTs are specified as $p_{m_i^{n,t+1}}(v|\vec{m}_i^t)$. With some abuse of notation, we also write $\Pr(m_i^{n,t+1}|\vec{m}_i^t, I_{\rightarrow i})$ for the probability of (some value of) a non-local factor $m_i^{n,t+1}$ according to $I_{\rightarrow i}$. Because the CPTs can only depend on $\vec{m}_i$, an incoming influence point $I_{\rightarrow i}$ enables the computation of a best response π_i independent of the other agents.

Of course, in general the actions of agent i can also influence other agents, so in order to find optimal solutions, we will also need to reason about this influence. We denote by $I_{i\rightarrow}$ the *outgoing influence point* of agent i, which specifies a collection of CPTs: one for each of its locally affected MMFs. Again, these CPTs can depend on only (the history) of MMFs $\vec{m}_i$. An incoming and outgoing influence point together form a (complete) *influence point* $I_i = \langle I_{\rightarrow i}, I_{i\rightarrow} \rangle$. A *joint influence point* $I = \langle I_{1\rightarrow}, \ldots, I_{n\rightarrow} \rangle$ specifies an outgoing influence point for each agent. Note that I also specifies the incoming influences, since every incoming influence point is specified by the outgoing influence points of the other agents. Fig. 3 illustrates the dependencies of an influence point in a so-called *influence DBN*. For instance, the possi-



t	$\vec{l}_{target}^t$	f_1^t	f_1^{t+1}	Pr
⋮	⋮	⋮	⋮	⋮
2	⟨B, A, A⟩	not found	found	0.9
⋮	⋮	⋮	⋮	⋮

$\vec{l}_{target}^t$
f_1^t
f_2^t
f_1^{t+1}
f_2^{t+1}
$I_{1\rightarrow}^{t+1} = \Pr(f_1^{t+1}|f_1^t, \vec{l}_{target}^t) = p_{f_1^{t+1}}$
$I_{2\rightarrow}^{t+1} = \Pr(f_2^{t+1}|f_2^t, \vec{l}_{target}^t) = p_{f_2^{t+1}}$

Figure 3: The Influence DBN for HouseSearch.

[2]Note that m_i^t is contained in s_i^t such that we can write $b_i(s_i^t, \vec{m}_i^{t-1}) = b_i(x_i^t, \vec{m}_i^t)$.

[3]For $t = 0$, the (non-conditional) distribution is specified by the initial state distribution b^0. The CPTs for subsequent stages may differ (from one another) because they summarize other agents' policies, which can depend on history.

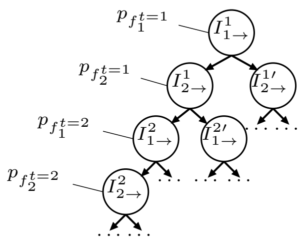


Figure 4: The influence search tree for HouseSearch.

ble CPTs $p_{f_1^{t+1}}$ are conditioned on $\vec{l}^t_{target}$, the history of the target location, as well as f^t_1, the value of 'target found by agent 1' at the previous stage.

Given I_i, agent i has an augmented local POMDP with local states, rewards and transitions. In this local model, a state is a pair $\langle s^t_i, \vec{m}^{t-1}_i \rangle$ (or equivalently $\langle x^t_i, \vec{m}^t_i \rangle$), such that, as discussed above, a belief is of the form $b_i(s^t_i, \vec{m}^{t-1}_i)$. Given an incoming influence point that dictates the transition probabilities of its nonlocally-affected MMFs, this local POMDP is independent of the other agents, but subject to the constraint that its solution must be a policy that adheres to the probabilities dictated by the outgoing influence point (specified by I_i). We call such a restricted model together with the influence point an *influence augmented local model (IALM)*. Solving the IALM is non-trivial since standard POMDP solvers will not respect the additional constraints. The problem can be solved by reformulating as a mixed integer linear program (MILP) [24, chapter 5].

2.4 Optimal Influence Search

The key property of these influences is that they can be used to compactly represent many of the other agents' policies. Rather than searching in the larger space of joint policies, we can search in the space of joint influence points and for each of them compute the agents' best responses to compute their value. In particular, the value of a fully specified joint influence point is:

$$V(I) = \sum_{i=1}^{n} V_i(I), \tag{1}$$

where $V_i(I) = V_i(\langle I_{\rightarrow i}, I_{i\rightarrow} \rangle)$ is the value of agent i's best response against $I_{\rightarrow i}$ subject to the constraints of satisfying $I_{i\rightarrow}$, i.e., the value that results from solving its IALM.

Given that we can compute the value of a joint influence point I, we can optimally solve a TD-POMDP by enumerating all I. Optimal Influence Search (OIS) [25] does this by constructing a tree, as illustrated in Fig. 4. An outgoing influence *slice* $I^t_{i\rightarrow}$ is that part of agent i's outgoing influence point corresponding to a particular stage t. The search tree contains the outgoing influence slices for all agents for stage $t = 1$ on the first n levels, it contains the slices for $t = 2$ on the next n levels, etc. An influence point is defined by a complete path from root to leaf. OIS performs an exhaustive depth-first search to find the optimal joint influence point from which the optimal joint policy can be reconstructed.

Although an apparently simple search strategy, OIS in fact demonstrated that influence abstraction can lead to significant gains in performance, thereby establishing itself as a state-of-the-art method for computing optimal solutions for weakly-coupled agents [25].

3. HEURISTIC INFLUENCE SEARCH

The previous section explained how OIS can greatly improve over other methods by searching in the search of joint influences, which can be much smaller than the space of joint policies. However, the weakness of OIS is that it needs to search this space exhaustively. In contrast, for general Dec-POMDPs, heuristic search methods (in particular A*, see, e.g., [17]) have shown to be very effective [19]. The main idea here, therefore, is to extend heuristic search to be able to search over the joint influence space.

In the subsections that follow, we develop the mechanics necessary to compute admissible heuristic values for nodes of the influence search tree. As we describe, this is a non-trivial extension, due to the fact that an influence summarizes a set of possible policies.

3.1 Computing Heuristic Values

To guarantee that heuristic search finds the optimal solution we need an *admissible* heuristic; i.e, a function F mapping nodes to heuristic values that are guaranteed to be an over-estimation of the value of the best path from root to leaf that passes through that node. In our setting this means that the heuristic $F(\check{I})$ for a partially specified joint influence point $\check{I}$ (corresponding to a path from the root of the tree to a non-leaf node) should satisfy

$$F(\check{I}) \geq \max_{I \text{ consistent with } \check{I}} V(I). \tag{2}$$

We will also write $I^{*|\check{I}}$ for the maximizing argument of the r.h.s. of (2).

In Dec-POMDPs, it is possible to perform A* search over partially specified joint policies [14]. For a 'past joint policy' $\boldsymbol{\varphi} = (\pi^0, \ldots, \pi^{t-1})$ that specifies the joint policy for the first t stages, it is possible to define $F(\boldsymbol{\varphi}) = G(\boldsymbol{\varphi}) + H(\boldsymbol{\varphi})$, where G gives the actual expected reward over the first t stages $0, \ldots, (t-1)$ and where H is a heuristic of the value achievable for the remaining stages. There are multiple ways to define H. For instance, one general form [20] is:

$$H(\boldsymbol{\varphi}) = \sum_{s} \Pr(s|\boldsymbol{b}^0, \boldsymbol{\varphi}) H^t(s), \tag{3}$$

where $H^t(s)$ is a guaranteed overestimation of the expected value starting from s in stage t. Such an overestimation can be obtained, for instance, by solving the underlying MDP (called Q_{MDP}) or POMDP [5, 14] .

Unfortunately, it is not possible to adapt the above approach to searching influence space in a straightforward fashion. Given an $\check{I}$, the past joint policy is not fixed, because $\pi^{*|\check{I}}$ the best joint policy for $I^{*|\check{I}}$ is unknown. Therefore, we take a somewhat different approach, as detailed next.

3.2 Restricted Scope Restricted Horizon

We exploit the fact that $V(I)$ in (1) can be additively decomposed. That is we upper bound (1) by:

$$F(\check{I}) = \sum_{i=1}^{n} F_i(\check{I}). \tag{4}$$

Clearly, when $F_i(\check{I}) \geq V_i(I^{*|\check{I}})$ for all agents i, then $F(\check{I}) \geq V(I^{*|\check{I}})$ and $F(\check{I})$ is admissible.

The problem of computing a heuristic value $F_i(\check{I})$ is illustrated in Figure 5. It shows that for a certain node $\check{I}$ in the search tree, the influences for the first number ($\overline{h}$) of stages

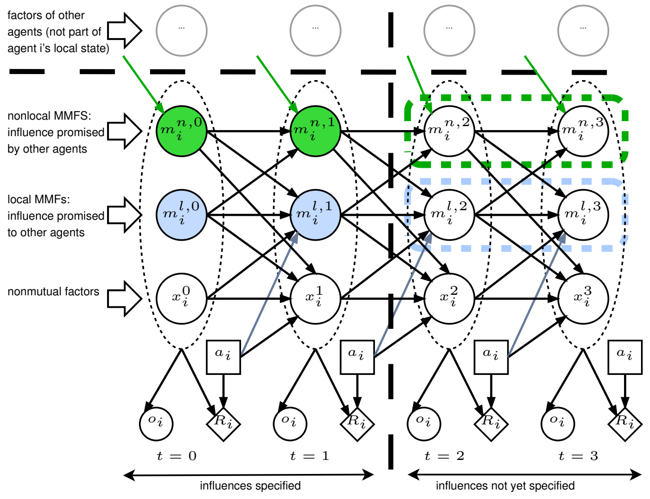

factors of other agents (not part of agent i's local state)
nonlocal MMFS: influence promised by other agents
local MMFs: influence promised to other agents
nonmutual factors
$t = 0$
$t = 1$
$t = 2$
$t = 3$
influences specified
influences not yet specified

Figure 5: A partially specified joint influence point $\check{I}$ from the perspective of agent i. Dashed black elipses denote the agent's local state. The figure does not include unaffectable factors. Influences are specified for stage 0,1. Green (dark) nodes are specified incoming influences, blue (light) nodes are specified outgoing influences. The dashed boxes denote the unspecified incoming (green) and outgoing (blue) influences for stages 2,3.

are specified (up to but *not* including stage $\overline{h}$). For now we assume that *all* influences at stage $\overline{h}-1$ are specified, i.e., we assume that $\check{I}^{\overline{h}-1}$ is a fully specified influence slice. Figure 5, in which $\overline{h}=2$, shows that computation of $F_i(\check{I})$ depends on only a subset of state factors (i.e., a restricted scope). In order to actually compute the $F_i(\check{I})$, we suggest a 2-step approach: 1) compute an admissible heuristic for the stages for which the influence is not yet specified, and 2) subsequently use these heuristic values to solve a constrained POMDP over horizon $\overline{h}$. We will refer to heuristics of this form as *restricted scope restricted horizon (RSRH)* heuristics.

3.2.1 Step 1: The Unspecified-Influence Stages.

The goal here is to, for each IALM state, to compute a heuristic value $H_i^{\overline{h}}$, analogous to the term used in (3), that is an optimistic estimate of the value of that state over the remaining (unspecified-influence) stages. In particular, we use an approach similar to Q_{MDP}: we compute the value of the underlying MDP but restricted to local states of the agent. In order to do so, we make optimistic assumptions on the unspecified incoming transition influences. Intuitively, this amounts to assuming that an agent i's peers will adopt policies that will exert the most beneficial effect on agent i's local state.

Remember that an IALM state $\langle s_i^t, \vec{m}_i^{t-1}\rangle = \langle x_i^t, \vec{m}_i^t\rangle$, and that we write $x_i = \langle x_i^l, x_i^u\rangle$ and $m_i = \langle m_i^l, m_i^n, m_i^u\rangle$. Now the overestimation we use is:

$$H_i^t(x_i, \vec{m}_i) \triangleq \max_{a_i}\Big[R(s_i,a_i) + \sum_{x_i', m_i^{l\prime}, m_i^{u\prime}} \Pr(x_i', m_i^{l\prime}, m_i^{u\prime}|s_i, a_i)\max_{m_i^{n\prime}} H_i^{t+1}(x_i', \vec{m}_i')\Big], \quad (5)$$

which upper bounds the value of the underlying restricted-scope MDP given *any* incoming influence point $I_{\rightarrow i}$:

$$V_{i,MDP}^{I_{\rightarrow i}}(x_i, \vec{m}_i) = \max_{a_i}\Big[R(s_i,a_i) + \sum_{x_i', m_i^{l\prime}, m_i^{u\prime}, m_i^{n\prime}} \Pr(x_i', m_i^{l\prime}, m_i^{u\prime}, m_i^{n\prime}|x_i, \vec{m}_i, a_i, I_{\rightarrow i}) V_{i,MDP}^{I_{\rightarrow i}}(x_i', \vec{m}_i')\Big]. \quad (6)$$

Also, it is important to note that $\Pr(x_i', m_i^{l\prime}, m_i^{u\prime}|s_i, a_i)$ in (5) can be directly computed due to the structure imposed by the TD-POMDP. As such, our optimistic estimate H_i^t can be computed via dynamic programming starting at the last stage $h-1$ and working back to stage $\overline{h}$.

3.2.2 Step 2: The Specified-Influence Stages.

Here we use $H_i^{\overline{h}}$ found in stage 1 to construct a restricted-horizon constrained POMDP, i.e., the IALM for agent i for only the first $\overline{h}$ stages, which we will denote by $\overline{M}$ (we denote all quantities of $\overline{M}$ with bars). For this IALM, we change the immediate rewards for the 'last' stage, stage $\overline{h}-1$, to include the heuristic $H_i^{\overline{h}}$ for the remaining stages:

$$\overline{R}^{\overline{h}-1}(x_i, \vec{m}_i, a_i) \triangleq R(s_i,a_i) + \sum_{x_i', m_i^{l\prime}, m_i^{u\prime}} \Pr(x_i', m_i^{l\prime}, m_i^{u\prime}|s_i, a_i)\max_{m_i^{n\prime}} H_i^{\overline{h}}(x_i', \vec{m}_i'). \quad (7)$$

That is, we apply the same optimistic estimate, effectively transforming the immediate rewards of stage $\overline{h}-1$ into optimistic heuristic 'action-value' estimates. The result is a completely specified, restricted-horizon, IALM for agent i that can be solved in exactly the same way as the full-horizon IALM. The value it achieves is $F_i(\check{I}) \triangleq \overline{V}_i(\overline{I})$.

3.2.3 Partially Specified Joint Influence Slices.

So far we assumed that the (outgoing) influences, for all agents, up to and including stage $\overline{h}-1$ were specified. However, for many nodes in the influence tree in Figure 4 the influences are only specified for a subset of agents at stage $\overline{h}-1$. However, we can easily overcome this problem by adapting the computation of $F_i(\check{I})$ in the following fashion.

If an outgoing influence at stage $\overline{h}-1$ is not specified we just omit the constraint in the MILP. If an incoming influence at stage $\overline{h}-1$ is not specified we transform the transition probability for the last transition in the restricted-horizon IALM (i.e., the transition from stage $\overline{h}-2$ to $\overline{h}-1$) such that for all $\langle x_i^{l,\overline{h}-1}, x_i^{u,\overline{h}-1}, m_i^{l,\overline{h}-1}, m_i^{u,\overline{h}-1}\rangle$ the local state will always transition to the fully specified local state $\langle x_i^{l,\overline{h}-1}, x_i^{u,\overline{h}-1}, m_i^{l,\overline{h}-1}, m_i^{n,\overline{h}-1}, m_i^{u,\overline{h}-1}\rangle$ with the highest heuristic value.

THEOREM 1. *$F_i(\check{I})$ is admissible.*

The implication of Theorem 1, whose proof is given in the appendix, is that our heuristic can be used to prune those influence assignments that are guaranteed to be suboptimal. As such, we will be able to expand potentially far fewer nodes of the influence-space search tree and still guarantee optimality.

3.3 A Tighter Heuristic

While the heuristic of the previous section is admissible, it is not very tight, because the second maximization in (5) corresponds to *always* assuming the most optimistic incoming influences. For instance, in the rectangle example, it will assume that the other agent finds the target in the second stage $t = 1$. However, from the possible locations of the target, we know that it will never be possible for the other agent to find the target at $t = 1$, it will take at least two steps. Next, we present a new heuristic that exploits this insight to yield a tighter upper bound to use during search.

Note that $V^{I_{\to i}}_{i,MDP}(x^t_i, \vec{m}^t_i)$ in (6) can be expanded to

$$\max_{a_i} \Big[R_i(s_i, a_i) + \sum_{\langle x_i, m^l_i, m^u_i\rangle^{t+1}} \Pr(\langle x^t_i, m^l_i, m^u_i\rangle^{t+1} | x^t_i, m^t_i, a_i) \sum_{m^{n,t+1}_i} \Pr(m^{n,t+1}_i | \vec{m}^t_i, I_{\to i}) V^{I_{\to i}}_{i,MDP}(x^{t+1}_i, \vec{m}^{t+1}_i) \Big] \quad (8)$$

due to the structure of the TD-POMDP. An important aspect in (8) is that $\Pr(m^{n,t+1}_i | \vec{m}^t_i, I_{\to i})$ exactly corresponds to one of the entries in $p_{m^{n,t+1}_i}$ that $I_{\to i}$ specifies.

Our first heuristic picks the heuristic best values for m^n_i, which we will denote $m^{n\star}_i$, and then assumes a optimistic influence $I^\star_{\to i}$ that prescribes $\Pr(m^{n\star}_i | \vec{m}^t_i, I^\star_{\to i}) = 1$. Here the idea is to use a more realistic (but still optimistic) influence $I^*_{\to i}$ by making use of an upper bound on $\Pr(m^{n\star}_i | \vec{m}^t_i, I_{\to i})$, which may be precomputed by examining the TD-POMDP CPT for factor m^n_i.

In particular, we compute the following upper bounds on $p_{m^{n,t}_i = v}$ the probability of each value v of a non-local factor $m^{n,t}_i$ as follows:

$$UB_{m^{n,t}_i | \vec{m}^{t-1}_i}(v) = \max_{\mathbf{v}_p \in PPP(m^{n,t}_i | \vec{m}^{t-1}_i)} \Pr(v | \mathbf{v}_p), \quad (9)$$

where $\Pr(\cdot|\cdot)$ is specified by a CPT of the 2TBN, $\mathbf{v}_p$ denotes an instantiation of the parents of $m^{n,t}_i$ in the 2TBN, and where the positive probability parents, $PPP(m^{n,t}_i | \vec{m}^{t-1}_i)$, is the set of such instantiations that 1) have positive probability of occuring, and 2) are consistent with $m^{n,t-1}_i$ (specified by $\vec{m}^{t-1}_i$).

We now use these bounds to define the more realistic influence $I^*_{\to i}$ and thus heuristic. In order to compute heuristic value $H(x^{t-1}_i, \vec{m}^{t-1}_i)$, we first order the possible values of $m^{n,t}_i$ according to their heuristic next-stage value $H(x^t_i, \vec{m}^t_i)$. Next, we define $I^*_{\to i}$ to be such that it defines a CPT $p_{m^{n,t}_i}$ that gives maximal weight to the high-ranked values of $m^{n,t}_i$. We create this distribution as follows: first we select the highest ranked value v^* of $m^{n,t}_i$ and we assign it probability $UB_{m^{n,t}_i | \vec{m}^{t-1}_i}(v^*)$, then we select the next best ranked value v' and either assign it the remaining probability mass (if that is less then its upper bound) or we assign it its upper bound and continue with the next-best value, etc. The heuristic is now defined by substituting the thusly obtained distribution for $\Pr(m^{n,t+1}_i | \vec{m}^t_i, I_{\to i})$ in (8).

4. EXPERIMENTS

We now present a empirical evaluation of our heuristic influence-space search method. Our primary hypothesis is that exhaustive optimal influence-space search (OIS), which has been shown to solve a number of weakly-coupled transition-dependent problems more efficiently than policy space search methods, can gain even more traction if combined with heuristic search methods. Although it would be interesting to additionally compare with optimal Dec-POMDP solution methods that employ heuristic search but not influence abstraction (e.g., [19]), we expect that the problems considered here are too large, especially in the number of individual observations ($4 \times 2 \times 2 = 16$ for *Diamond*, 32 for *Rectangle*, and 36 for *Squares*), which are beyond what optimal Dec-POMDP solvers have demonstrated to handle (the largest of those problems have 5 individual observations). In order to test our hypothesis, we performed experiments both on the HouseSearch configurations shown in Fig. 1 as well as on SatelliteRover, a TD-POMDP test set involving two agents that interact through task dependencies (we use the version where the agents can wait) [25].

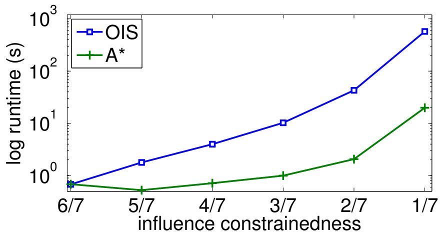


Figure 6: Runtimes on SatelliteRover.

For HouseSearch, we experimented with different degrees of stochasticity. I.e., we considered problems ranging from deterministic observations and deterministic actions, labeled “d.o.d.a.”, to stochastic observations (where the probability of observing no target when in the same room as the target is 0.25) and stochastic actions (where the probability that a move action will fail is 0.1), labeled “s.o.s.a”. For all problems, the parameters were set to $c_{time} = -5$, $c_i = -1$ for each movement action, and $r_{detect} = 0$. Table 2 compares the runtimes of OIS with those of A* using the two variants of our restricted scope restricted horizon heuristic (where A*$_1$ corresponds to that described in Section 3.2 and A*$_2$ corresponds to that described in Section 3.3). As shown, using the simplest variant of our heuristic can lead to significant speed-ups over depth-first search, especially on the *Diamond* configuration where we see as much as two orders of magnitude improvement (e.g., at horizon 3 of *Diamond* s.o.s.a). It also allows scaling up to larger time horizons than was previously possible. We also see that the heuristic A*$_2$ is indeed tighter, allowing for more pruning and hence faster solutions on almost all problems. For *Rectangle* and *Squares*, however, the benefit of A* over exhaustive OIS are less pronounced. (Given space restrictions, we omit the d.o.d.a., d.o.s.a, and s.o.s.a. variations of these problems, whose trends were the same as in s.o.d.a.)

We also tested A*$_1$ on SatelliteRover, in which the lengths of task execution windows were systematically varied to affect the level of *influence constrainedness* [25]. The less constrained the agents' interactions, the larger the influence space, as demonstrated by the exponentially increasing runtimes plotted on a logarithmic scale in Fig. 6. Evidently, it is on these less-constrained problems, which are hardest for OIS, where we get the most speedup from A*. Here, A* search leads to significant savings of well over an order of magnitude (573s vs. 19.9s for IC=1/7), thereby complementing the savings achieved by influence-based abstraction.

The differences between the impact of A* in *Diamond*, *Rectangle*, and *Squares* warrant a more detailed analysis. In

	Diamond (d.o.d.a)			*Diamond* (s.o.d.a)			*Diamond* (d.o.s.a)			*Diamond* (s.o.s.a)			*Rectangle* (s.o.d.a)			*Squares* (s.o.d.a)		
h	OIS	A^*_1	A^*_2	OIS	A^*_1	A^*_2	OIS	A^*_1	A^*_2	OIS	A^*_1	A^*_2	OIS	A^*_1	A^*_2	OIS	A^*_1	A^*_2
1	0.28	0.27	0.29	0.19	0.30	0.25	0.22	0.28	0.23	0.25	0.32	0.23	0.08	0.10	0.17	0.21	0.25	0.28
2	1.64	0.83	0.26	2.28	0.92	0.25	3.13	1.86	0.29	8.68	2.41	0.64	0.72	0.71	0.47	1.73	1.33	0.67
3	8.84	1.77	0.68	35.60	5.93	0.89	151.6	11.63	1.38	8,871	52.08	2.37	16.52	17.88	7.85	31.96	34.55	10.22
4	101.6	8.50	1.28	811.4	48.75	2.86		436.0	4.89		3,066	14.39	621.2	412.3	138.6	1,716	1,101	167.5
5	945.0	31.80	7.90		953.1	44.57			178.1			44.52			4,187			6,295

Table 2: Runtimes (in seconds), including heuristic computation (observed to be negligible), on variations of HouseSearch.

the latter two variations, the tighter heuristic appears too loose to effectively guide heuristic search except on problems with longer time horizons. Upon closer inspection, we discovered an inherent bias in the application of our heuristic to HouseSearch problems; it encourages the 'stay' action. This is because the heuristic evaluation of each agent makes the optimistic assumption that the other agent will probably find the target, in which case the agent need not look itself and incur the associated movement cost. To remedy this problem, we developed a simple specialized adaptation (A*-*imc*) that ignores the movement cost component of the factored reward in the first term of Equation 7. While this modification causes the heuristic to be less tight, it also takes away the bias against movement actions. Results for this modification are shown in Table 3. An interesting phenomenon occurs for *Rectangle* and *Squares* where for longer time horizons, runtimes are significantly decreased because the no-movement bias has been eliminated, but where for for shorter horizons we see slight increases in runtimes because here the optimal policy is actually to stay. Likewise, for *Diamond*, very little movement is required to find the target; in this case, the search also suffers from the fact that ignoring movement costs actually loosens the heuristic, causing more nodes to be expanded. All in all, this demonstrates that using specialized domain knowledge can significantly increase the effectiveness of A* influence space search.

	Diamond (s.o.d.a)		*Rectangle*		*Squares*	
h	A^*_2	A^*_2-*imc*	A^*_2	A^*_2-*imc*	A^*_2	A^*_2-*imc*
1	0.25	0.56	0.17	0.30	0.28	0.50
2	0.25	0.34	0.47	0.57	1.67	1.20
3	0.89	1.10	7.85	7.18	10.22	12.10
4	2.86	3.86	138.6	14.71	167.5	46.95
5	44.57	146.5	4,187	222.0	6,295	422.6

Table 3: Ignoring movement costs in heuristic calculation.

5. RELATED WORK

Having reviewed Dec-POMDP heuristic search [14, 19, 20] and TD-POMDP influence-space search [24, 25], which are most closely related to the work we have developed here, we now describe connections to other recent models and methods. For instance, the EDI-CR model [9] makes explicit a set of joint transition and reward dependencies. The authors propose an MILP-based solution method that is conceptually related to influence abstraction; it clusters action-observation histories that have equivalent probabilistic effects so as to reduce the number of joint histories considered. A significant difference is that, unlike the algorithms we develop here, instead, it entails solving a single joint model framed as an MILP, instead of decoupling the problem into influence-abstracted local models.

The DPCL [22, 23] exploits 'coordination locales' for efficient computation of an agent's response policy by incorporating effects of other agents' policies into a compact local model, which is similar the TD-POMDP's IALM. However, in contrast to the optimal search methods that we develop, the DPCL has only ever afforded approximate solutions. Other methods have been developed for computing approximate solutions for general factored Dec-POMDPs, of which the TD-POMDP could be considered a specialized instance. Their more general factored structure has been exploited by using collaborative graphical Bayesian games in combination with non-serial dynamic programming [15] and approximate inference [12] in the finite-horizon case. In the infinite-horizon case finite state controllers and EM [8, 16] have been proposed. In contrast to the work presented here, these methods search in the policy space rather than the influence space.

6. CONCLUSIONS & FUTURE WORK

We have introduced heuristic A* search of the influence space for the optimal solution of multiagent planning problems formalized as TD-POMDPs. As previous work has shown, the space of influences can be much smaller than the space of joint policies and therefore searching the former can lead to significant improvements in performance. We illustrated the efficacy of our approach on a optimal decentralized probabilistic search problem, thereby showing the first application of influence search on TD-POMDPs with cyclic dependencies between agents. Our empirical evaluation shows that A* search of the influence space can lead to significant improvements in performance over exhaustive OIS. In particular, the results indicate that in problems that are harder (i.e., where there is a high number of possible influences) A* leads to the most improvements. In other words, influence abstraction and heuristic search can provide complementary gains. This suggests that A* search of influence space can be an important tool in scaling up a large class of multiagent planning problems under uncertainty.

There are a number of directions for future research. Because of the connection this paper establishes between searching influence space and MAA* for Dec-POMDPs, it is natural to try and extend recent improvements in the latter to the former. One question is whether it is possible to incrementally expand the nodes in the search tree. Such incremental expansion has yielded significant increases in performance for Dec-POMDPs [19]. Another interesting question is whether it is possible to cluster influence points. That is, it may be possible to characterize when different joint influence points correspond to best responses that are guaranteed to be the identical.

7. ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their constructive feedback. This work was supported, in part, by the Fundação para a Ciência e a Tecnologia (FCT) and the CMU-Portugal Program under project CMU-PT/SIA/0023/2009, by AFOSR MURI project #FA9550-09-1-0538, and by NWO CATCH project #640.005.003.

8. REFERENCES

[1] R. Becker, S. Zilberstein, and V. Lesser. Decentralized Markov decision processes with event-driven interactions. In *AAMAS*, pages 302–309, 2004.

[2] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Solving transition independent decentralized Markov decision processes. *JAIR*, 22:423–455, 2004.

[3] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Math. of OR*, 27(4):819–840, 2002.

[4] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *JAIR*, 11:1–94, 1999.

[5] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *AAMAS*, pages 136–143, 2004.

[6] M. Hauskrecht. Value-function approximations for partially observable Markov decision processes. *JAIR*, 13:33–94, 2000.

[7] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

[8] A. Kumar, S. Zilberstein, and M. Toussaint. Scalable multiagent planning using probabilistic inference. In *IJCAI*, pages 2140–2146, 2011.

[9] H. Mostafa and V. Lesser. Compact mathematical programs for DEC-MDPs with structured agent interactions. In *UAI*, pages 523–530, 2011.

[10] R. Nair, M. Tambe, M. Yokoo, D. V. Pynadath, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *IJCAI*, 2003.

[11] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *AAAI*, pages 133–139, 2005.

[12] F. A. Oliehoek. *Value-Based Planning for Teams of Agents in Stochastic Partially Observable Environments*. PhD thesis, University of Amsterdam, 2010.

[13] F. A. Oliehoek, J. F. Kooi, and N. Vlassis. The cross-entropy method for policy search in decentralized POMDPs. *Informatica*, 32:341–357, 2008.

[14] F. A. Oliehoek, M. T. J. Spaan, and N. Vlassis. Optimal and approximate Q-value functions for decentralized POMDPs. *JAIR*, 32:289–353, 2008.

[15] F. A. Oliehoek, M. T. J. Spaan, S. Whiteson, and N. Vlassis. Exploiting locality of interaction in factored Dec-POMDPs. In *AAMAS*, pages 517–524, 2008.

[16] J. Pajarinen and J. Peltonen. Efficient planning for factored infinite-horizon DEC-POMDPs. In *IJCAI*, pages 325–331, 2011.

[17] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2nd edition, 2003.

[18] S. Seuken and S. Zilberstein. Memory-bounded dynamic programming for DEC-POMDPs. In *IJCAI*, 2007.

[19] M. T. J. Spaan, F. A. Oliehoek, and C. Amato. Scaling up optimal heuristic search in Dec-POMDPs via incremental expansion. In *IJCAI*, pages 2027–2032, 2011.

[20] D. Szer, F. Charpillet, and S. Zilberstein. MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *UAI*, pages 576–583, 2005.

[21] P. Varakantham, J. Marecki, Y. Yabu, M. Tambe, and M. Yokoo. Letting loose a SPIDER on a network of POMDPs: Generating quality guaranteed policies. In *AAMAS*, 2007.

[22] P. Varakantham, J. young Kwak, M. Taylor, J. Marecki, P. Scerri, and M. Tambe. Exploiting coordination locales in distributed POMDPs via social model shaping. In *ICAPS*, 2009.

[23] P. Velagapudi, P. Varakantham, P. Scerri, and K. Sycara. Distributed model shaping for scaling to decentralized POMDPs with hundreds of agents. In *AAMAS*, 2011.

[24] S. J. Witwicki. *Abstracting Influences for Efficient Multiagent Coordination Under Uncertainty*. PhD thesis, University of Michigan, 2011.

[25] S. J. Witwicki and E. H. Durfee. Influence-based policy abstraction for weakly-coupled Dec-POMDPs. In *ICAPS*, 2010.

APPENDIX

PROOF OF THEOREM 1. We need to show that

$$\forall_{I_i} \quad F_i(\check{I}) = \overline{V}_i(\overline{I}) \geq V_i(I^{*|\check{I}}) \tag{10}$$

We assume an arbitrary $I_{\rightarrow i}, I_{i\rightarrow}$ consistent with $\check{I}$. Since the first $\overline{h}-1$ stages are identical, (10) clearly holds if

$$\forall_{b_i,a_i} \qquad \overline{Q}_i^{\overline{h}-1}(b_i,a_i) \geq Q_i^{\overline{h}-1,I_{\rightarrow i},I_{i\rightarrow}}(b_i,a_i). \tag{11}$$

We choose an arbitrary b_i,a_i. Expanding both sides, we need to show that

$$\overline{R}^{\overline{h}-1}(b_i,a_i) \geq R^{\overline{h}-1}(b_i,a_i) + \sum_{b'} P(b'|b_i,a_i) V_i^{\overline{h},I_{\rightarrow i},I_{i\rightarrow}}(b'_i). \tag{12}$$

Expanding the expectations over IALM states:

$$\sum_{x_i,\vec{m}_i} b_i(x_i,\vec{m}_i) \overline{R}^{\overline{h}-1}(x_i,\vec{m}_i,a_i) \geq \sum_{x_i,\vec{m}_i} b_i(x_i,\vec{m}_i) \Big[R(s_i,a_i) + \sum_{s'_i}\sum_{o_i} \Pr(s'_i,o_i|s_i,a_i) V_i^{\overline{h},I_{\rightarrow i},I_{i\rightarrow}}(x'_i,\vec{m}'_i,b'_i).\Big]$$

Substituting the definition of $\overline{R}$:

$$\sum_{x_i,\vec{m}_i} b_i(x_i,\vec{m}_i)\Big[R(s_i,a_i) + \sum_{x'_i,m_i^{l\prime},m_i^{u\prime}} \Pr(x'_i,m_i^{l\prime},m_i^{u\prime}|s_i,a_i) \max_{m_i^{n\prime}} H_i^{\overline{h}}(x'_i,\vec{m}'_i)\Big] \geq \sum_{x_i,\vec{m}_i} b_i(x_i,\vec{m}_i)\Big[R(s_i,a_i) + \sum_{s'_i}\sum_{o_i} \Pr(s'_i,o_i|s_i,a_i) V_i^{\overline{h},I_{\rightarrow i},I_{i\rightarrow}}(x_i,\vec{m}_i,b'_i)\Big].$$

This is proven if we can show that

$$\forall_{x_i,\vec{m}_i} \quad \sum_{x'_i,m_i^{l\prime},m_i^{u\prime}} \Pr(x'_i,m_i^{l\prime},m_i^{u\prime}|s_i,a_i) \max_{m_i^{n\prime}} H_i^{\overline{h}}(x'_i,\vec{m}'_i) \geq \sum_{s'_i}\sum_{o_i} \Pr(s'_i,o_i|s_i,a_i) V_i^{\overline{h},I_{\rightarrow i},I_{i\rightarrow}}(x'_i,\vec{m}'_i,b'_i) \tag{13}$$

We assume arbitrary $x_i, \vec{m}_i$ and now continue with the right hand side. Since it is well-known that the MDP value function is an upper bound to the POMDP value function [6], we have

$$\begin{aligned} & \sum_{s'_i}\sum_{o_i} \Pr(s'_i,o_i|s_i,a_i) V_i^{\overline{h},I_{\rightarrow i},I_{i\rightarrow}}(x'_i,\vec{m}'_i,b'_i) \\ \leq \; & \sum_{s'_i}\sum_{o_i} \Pr(s'_i,o_i|s_i,a_i) V_{i,MDP}^{\overline{h},I_{\rightarrow i},I_{i\rightarrow}}(x'_i,\vec{m}'_i) \\ \leq \; & \sum_{s'_i} \Pr(s'_i|s_i,a_i) V_{i,MDP}^{\overline{h},I_{\rightarrow i},I_{i\rightarrow}}(x'_i,\vec{m}'_i) \\ \leq \; & \sum_{s'_i} \Pr(s'_i|s_i,a_i) V_{i,MDP}^{\overline{h},I_{\rightarrow i}}(x'_i,\vec{m}'_i) \end{aligned} \tag{14}$$

The last term denotes the optimal value under only incoming influences, and the inequality holds because the set of policies available to agent i without restrictions due to promised outgoing influences is a strict superset of those when there are outgoing influences. Now, by (6) we directly get that the last quantity

$$\leq \sum_{s'_i} \Pr(s'_i|s_i,a_i) H_i^{\overline{h}}(x'_i,\vec{m}'_i) \leq \sum_{x'_i,m_i^{l\prime},m_i^{u\prime}} \Pr(x'_i,m_i^{l\prime},m_i^{u\prime}|s_i,a_i) \max_{m_i^{n\prime}} H_i^{\overline{h}}(x'_i,\vec{m}'_i), \tag{15}$$

which concludes the proof. □

A Hierarchical Goal-Based Formalism and Algorithm for Single-Agent Planning

Vikas Shivashankar[1] Ugur Kuter[2] Dana Nau[1] Ron Alford[1]
[1]University of Maryland, College Park, Maryland 20742 USA
[2]Smart Information Flow Technologies, Minneapolis, Minnesota 55401 USA
svikas@cs.umd.edu ukuter@sift.net nau@cs.umd.edu ronwalf@cs.umd.edu

ABSTRACT

Plan generation is important in a number of agent applications, but such applications generally require elaborate *domain models* that include not only the definitions of the actions that an agent can perform in a given domain, but also information about the most effective ways to generate plans for the agent in that domain. Such models typically take a large amount of human effort to create.

To alleviate this problem, we have developed a hierarchical goal-based planning formalism and a planning algorithm, GDP (Goal-Decomposition Planner), that combines some aspects of both HTN planning and domain-independent planning. For example, it allows the planning agent to use domain-independent heuristic functions to guide the application of both methods and actions.

This paper describes the formalism, planning algorithm, correctness theorems, and the results of a large experimental study. The experiments show that our planning algorithm works as well as the well-known SHOP2 HTN planner, using domain models only about half the size of SHOP2's.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Plan execution, formation, and generation*

General Terms

Algorithms

Keywords

AI planning, hierarchical planning, goal decomposition

1. INTRODUCTION

The ability to do effective planning is important for a wide variety of computerized agents. Examples include robotic agents (e.g., the Mars rovers [27]), game-playing agents (e.g., in card games [29, 25] and real-time strategy games [5]), web-service agents [19], and others. To build capable planners for agent environments, generally the planner must incorporate a *domain model* that includes not only the definitions of the basic actions that the agent can perform, but also information about the most effective ways to generate plans in the agent's environment. One way to incorporate domain models into planning agents is to custom-build a planning module for the application at hand. This approach was used successfully in many of the examples just mentioned, but it usually requires a huge development effort.

Another approach is to use a *domain-configurable* planner which reads the domain model as part of its input. Most such planners use *Hierarchical Task Network (HTN) planning*, in which the domain model includes *methods* for accomplishing *tasks* by dividing them into smaller and smaller subtasks. This approach has been used in a wide variety of planning domains, e.g., [33, 6, 22, 21, 26]. Writing a new domain model usually is much less work than building a new domain-specific planner, but in most cases it still requires a great deal of human effort.

We have developed a *Hierarchical Goal Network (HGN) planning* formalism and algorithm similar to the HTN formalism and algorithm in the SHOP planner [24], but with some important differences that make it easier to develop domain models. In the SHOP formalism, each task is a separate syntactic entity whose semantics depends entirely on what methods match it. In contrast, HGN tasks consist of *initial conditions* and *goal conditions* with the same semantics as in classical planning; and HGN methods and actions have applicability and relevance conditions similar to the ones for actions in classical planning. Our results are as follows:

Formalism: The HGN formalism provides (provably) as much expressive power as SHOP's HTN formalism (or equivalently, SHOP2's HTN formalism restricted to totally-ordered subtasks). Moreover, in contrast to problems with soundness (see Section 2) in HTN translations of classical planning domains, soundness is guaranteed for all HGN domain models of classical domains; and it is easier to analyze whether HGN translations are complete.

Planning algorithm: Our planning algorithm, *GDP (Goal Decomposition Planner)*, is sound and complete. It is similar in several ways to SHOP, but the HGN task and method semantics provide much more flexibility in applying methods and actions. For example, in writing GDP domain models we don't have to commit to a task name in the methods: we just specify what should be achieved instead of how to achieve it, and let the planner decide which methods and actions are relevant and applicable.

Heuristic function: The HGN task and method semantics enable the development of heuristic functions similar to the ones in classical planners (e.g., FF [16] and HSP [3]). We provide one such heuristic function, for optional use in GDP, to guide the selection of both methods and operators. For a domain model to work well, it is important to specify the order in which a set of methods and operators should be tried when more than one of them is applicable; and the heuristic function can make this easier by enabling the planner to deduce the best order on its own.

Experimental results: We have done an extensive experimen-

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

tal comparison of three versions of GDP, two versions of SHOP2, and FF, in five different planning domains. On average, GDP's domain models were only about half as large as the equivalent SHOP2 domain models, yet provided roughly the same performance (i.e., planning time and plan lengths). The HGN domain models were so much simpler because the HGN task and method semantics obviates the need for a plethora of extra methods and bookkeeping operations needed in SHOP2 domain models.

2. RELATED WORK

Over the years, several well-known researchers (e.g., [10, 17]) have argued for combining HTN planning with other techniques, and several of the older HTN planners (e.g., SIPE [33, 32] and O-PLAN [6, 30]) combined hierarchical decomposition with goal-directed partial-order planning.[1] But in Erol *et al*'s influential HTN formalism [8] and most subsequent HTN planning research (e.g., [14, 23, 22]; a notable exception is [20]), the definition of a solution is tied so intimately to HTN decomposition that the planning problems are solvable in no other way.[2]

The lack of correspondence between tasks and goals makes it hard to translate classical planning problems correctly into HTN domain models (e.g., a common error is to translate a goal $g_1 \wedge g_2$ into a task sequence $\langle \mathsf{achieve}(g_1), \mathsf{achieve}(g_2)\rangle$, ignoring the possibility that the plan for $\mathsf{achieve}(g_2)$ may delete the previously achieved goal g_1). In the 2000 International Planning Competition, SHOP [24] was disqualified because of an incorrect HTN translation that caused SHOP to return an incorrect answer. The lack of correspondence between tasks and goals has also interfered with recent efforts to combine HTN planning with classical planning [1, 13], necessitating several *ad hoc* modifications and restrictions.

3. FORMALISM

Classical planning. Following Ghallab et al. [14, Chap. 2]), we define a classical planning domain D as a finite state-transition system in which each state s is a finite set of ground atoms of a first-order language L, and each action a is a ground instance of a planning operator o. A planning operator is a triple $o = (\text{head}(o), \text{pre}(o), \text{eff}(o))$, where pre$(o)$ and eff(o) are sets of literals called o's *preconditions* and *effects*, and head(o) includes o's *name* and *argument list* (a list of the variables in pre(o) and eff(o)).

An action a is executable in a state s if $s \models \text{pre}(a)$, in which case the resulting state is $\gamma(a) = (s - \text{eff}^-(a)) \cup \text{eff}^+(a)$, where eff$^+(a)$ and eff$^-(a)$ are the atoms and negated atoms, respectively, in eff(a). A plan $\pi = \langle a_1, \ldots, a_n\rangle$ is executable in s if each a_i is executable in the state produced by a_{i-1}; and in this case we let $\gamma(s, \pi)$ be the state produced by executing the entire plan.

A *classical planning problem* is a triple $P = (D, s_0, g)$, where D is a classical planning domain, s_0 is the initial state, and g (the *goal formula*) is a set of ground literals. A plan π is a solution for P if π is executable in s_0 and $\gamma(s_0, \pi) \models g$.

HGN planning. An *HGN method* m has a head head(m) and preconditions pre(m) like those of a planning operator, and a sequence of subgoals sub$(m) = \langle g_1, \ldots, g_k\rangle$, where each g_i is a goal formula (a set of literals). We define the *postcondition* of m to be post$(m) = g_k$ if sub(m) is nonempty; otherwise post$(m) = $ pre(m).

An action a (or method instance m) is *relevant* for a goal formula g if eff(a) (or post(m), respectively) entails at least one literal in g and does not entail the negation of any literal in g.

Some notation: if $\pi_1, \ldots, \pi_n$ are plans or actions, then $\pi_1 \circ \ldots \circ \pi_n$ denotes the plan formed by concatenating them.

An *HGN planning domain* is a pair $D = (D', M)$, where D' is a classical planning domain and M is a set of methods. An *HGN planning problem* $P = (D, s_0, g)$ is like a classical planning problem except that D is an HGN planning domain. The set of solutions for P is defined recursively:

Case 1. If $s_0 \models g$, then the empty plan is a solution for P.

Case 2. Let a be any action that is relevant for g and executable in s_0. Let π be any solution to the HGN planning problem $(D, \gamma(s_0, a), g)$. Then $a \circ \pi$ is a solution to P.

Case 3. Let m be a method instance that is applicable to s_0 and relevant for g and has subgoals $g_1, \ldots, g_k$. Let π_1 be any solution for (D, s_0, g_1); let π_i be any solution for $(D, \gamma(s_0, (\pi_1 \circ \ldots \circ \pi_{i-1})), g_i)$, $i = 2, \ldots, k$; and let π be any solution for $(D, \gamma(s_0, (\pi_1 \circ \ldots \circ \pi_k)), g)$. Then $\pi_1 \circ \pi_2 \circ \ldots \circ \pi_k \circ \pi$ is a solution to P.

In the above definition, the relevance requirements in Cases 2 and 3 prevent classical-style action chaining unless each action is relevant for either the ultimate goal g or a subgoal of one of the methods. This requirement is analogous to (but less restrictive than) the HTN planning requirement that actions cannot appear in a plan unless they are mentioned explicitly in one of the methods. As in HTN planning, it gives an HGN planning problem a smaller search space than the corresponding classical planning problem.

The next theorem proves that HGN planning is *sound*: any HGN solution is also a solution to the corresponding classical problem.

THEOREM 1 (HGN SOUNDNESS). *Let $D = (D', M)$ be an HGN planning domain. For every (s_0, g), the set of solutions to the HGN planning problem $P = (D, s_0, g)$ is a subset of the set of solutions to the classical planning problem $P' = (D', s_0, g)$.*

PROOF. Let $\pi = \langle a_1, \ldots, a_n\rangle$ be any solution for P. From the definition of a solution, it follows that in the HGN domain D, π is executable in s_0 and $\gamma(s_0, \pi) \models g$. But $D = (D', M)$, so any action that is executable in D is also executable in the classical domain D' and produces the same effects. Thus it follows that in D', π is executable in s_0 and $\gamma(s_0, \pi) \models g$. □

THEOREM 2 (HGN COMPLETENESS). *For every classical planning domain D, there is a set of HGN methods M such that the classical planning problem $P = (D, s_0, g)$ and the HGN planning problem $P' = ((D, M), s_0, g)$ have the same set of solutions.*

PROOF. Let X be the set of all *simple* paths in D. For each path x in X, suppose M contains methods that will specify goals for each state on x as subgoals. Thus, each subgoal will be achieved by a single action such that when the sequence of actions applied from the start of x, and the result will be the end state. Then the theorem follows. □

The following two theorems prove that the HGN formalism provides expressive power equal to that of SHOP's HTN formalism:

THEOREM 3 (HTN EXPRESSIVITY). *For any HGN problem (D, s_0, g_0), there exists a totally-ordered HTN problem (D', s_0, t_{g_0}) such that (D, s_0, g_0) is solvable if and only if (D', s_0, t_{g_0}) is solvable.*

[1]PRS [12] is also a hierarchical goal-based reasoning system, but its primary focus is *reactive execution* in dynamic environments; the actual planning is rather limited.

[2]One might expect Erol *et al.*'s formalism to allow classical goal achievement, because it allows *goal tasks* of the form $\mathsf{achieve(goal)}$. But just as with any other task, a goal task's solution plans can only be constructed by HTN decomposition: the only difference is a constraint that the goal must be true after executing the plan.

Proof Sketch. We proceed to translate an HGN planning problem (D, s_0, g_0) into an HTN planning problem as follows: each goal formula g in D is represented by a task symbol t_g; g_0 is represented by the task symbol t_{g_0}. For each HGN method $\langle pre, \langle g_1, \ldots, g_k \rangle\rangle$, we create a new HTN method accomplishing task t_{g_k} with preconditions pre and subtasks $\langle t_{g_1}, \ldots, t_{g_k}\rangle$. Then for each t_g, we create an HTN method having a precondition of g and no subtasks. Also for every method or operator u relevant to g, we have a method accomplishing t_g having a precondition of $\neg g$ and subtasks $\langle t_u, t_g\rangle$, t_u being the task symbol corresponding to u. We then return the HTN problem $(D' \cup O, s_0, t_{g_0})$ where D' is the set of translated HTN methods and O is the set of planning operators.

It is now easy to show that any HGN decomposition trace can be mapped to a corresponding trace of the HTN problem thus constructed and vice-versa. Thus the theorem follows. □

THEOREM 4 (HGN EXPRESSIVITY). *For any totally-ordered HTN planning problem (D, s_0, t_0), there is an HGN planning problem (D', s_0, g_{t_0}) such that (D, s_0, t_0) is solvable if and only if (D', s_0, g_{t_0}) is solvable.*

Proof Sketch. To translate an HTN planning problem[3] (D, s_0, t_0), we create predicates $fin_t(.)$ for each task $t(.)$ to represent task completion. We add an extra predicate *lead* that is asserted by an artificial operator with no preconditions. We have artificial operators *assert-fin-t(.)* for each task symbol t that has precondition $\langle lead\rangle$ and effect $\langle \neg lead, fin_t(.)\rangle$ Each HTN method for task t with subtasks $\langle t_1, t_2, \ldots, t_n\rangle$ is now converted to an HGN method with the same preconditions and a sequence of subgoals $\langle fin_{t_1}, \neg fin_{t_1}, fin_{t_2}, \neg fin_{t_2}, \ldots fin_{t_n}, \neg fin_{t_n}, lead, fin_t\rangle$. The $\neg fin(.)$ subgoals are used to cleanup the state for future decompositions. The HGN planning problem is $(D' \cup O, s_0, fin_{t_0})$, where D' is the set of translated HGN methods and O is the set of classical planning operators and additional artificial operators described above. It is now easy to show that every HTN decomposition trace can be mapped to a corresponding trace of the HGN planning problem thus constructed and vice-versa. The theorem follows. □

The above theorems provide procedures to translate HGN planning problems to HTN problems and vice-versa in low-order polynomial time. This proves that HGN planning has the same expressive power as totally-ordered HTN planning.

Let HGN-PLAN-EXISTENCE be the following problem: *Given an HGN planning problem P, is there a plan that solves P?*

THEOREM 5. HGN-PLAN-EXISTENCE *is decidable.*

Proof Sketch. Erol et al. [9] prove that the plan existence problem for totally-ordered HTN planning is decidable. From this and Theorem 3, the result immediately follows. □

4. PLANNING ALGORITHM

Algorithm 1 is GDP, our HGN planning algorithm. It works as follows (where G is a stack of goal formulas to be achieved):

In Line 3, if G is empty then the goal has been achieved, so GDP returns π. Otherwise, GDP selects the first goal g in G (Line 4). If g is already satisfied, GDP removes g from G and calls itself recursively on the remaining goal formulae.

In Lines 7-8, if no actions or methods are applicable to s and relevant for g, then GDP returns failure. Otherwise, GDP nondeterministically chooses an action/method u from U.

[3]We assume a single task t_0 in the initial task network; this is without loss of generality as we can replace a totally-ordered initial task network with an artificial toptask and add an extra method decomposing the toptask to the initial task network.

Algorithm 1: A high-level description of GDP. Initially, D is an HGN planning domain, s is the initial state, g is the goal formula, $G = \langle g\rangle$, and π is $\langle\rangle$, the empty plan.

```
1  Procedure GDP($D, s, G, \pi$)
2  begin
3      if $G$ is empty then return $\pi$
4      $g \leftarrow$ the first goal formula in $G$
5      if $s \models g$ then
6          remove $g$ from $G$ and return GDP($D, s, G, \pi$)
7      $U \leftarrow \{$actions and method instances that are relevant
               for $g$ and applicable to $s\}$
8      if $U = \emptyset$ then return failure
9      nondeterministically choose $u \in U$
10     if $u$ is an action then
11         append $u$ to $\pi$ and set $s \leftarrow \gamma(s, u)$
12     else insert sub($u$) at the front of $G$
13     return GDP($D, s, G, \pi$)
14 end
```

If u is an action, then GDP computes the next state $\gamma(s, u)$ and appends u to π. Otherwise u is a method, so GDP inserts u's subgoals at the front of G. Then GDP calls itself recursively on G.

4.1 Formal Properties

The following theorems show that GDP is sound and complete:

THEOREM 6 (GDP SOUNDNESS). *Let $P = (D, s_0, g)$ be an HGN planning problem. If a nondeterministic trace of* GDP$(D, s_0, \langle g\rangle, \langle\rangle)$ *returns a plan π, then π is a solution for P.*

Proof Sketch. The proof is by induction on n, the length of π. When $n = 0$ (i.e. $\pi = \langle\rangle$), this implies that s_0 entails g. Hence, by Case 1 of the definition of a solution, π is a solution for P. Suppose that if GDP returns a plan π of length $k < n$, then π is a solution for P. At an invocation suppose GDP returns π of length n. The proof proceeds by showing the following. When GDP chooses an action or a method for the current goal at any invocation, then by induction, the plans returned from those calls are solutions to the HGN planning problems in those calls. Hence, by definition of solutions for P, π is a solution for P. □

THEOREM 7 (GDP COMPLETENESS). *Let $P = (D, s_0, g)$ be an HGN planning problem. If π is a solution for P, then a nondeterministic trace of* GDP$(D, s_0, \langle g\rangle, \langle\rangle)$ *will return π.*

Proof Sketch. The proof is by induction on n, the length of π. When $n = 0$, this implies that the empty plan is a solution for P and that $s_0 \models g$. Hence GDP would return $\langle\rangle$ as a solution. Suppose that if P has a solution of length $k < n$, then GDP will return it. At any invocation, the proof proceeds to show by induction the following. If GDP chooses an action a, then one of the nondeterministic traces of the subsequent call to GDP must return $\pi = a \circ \pi'$ where π' is a solution for the problem $P' = (D, \gamma(s_0, a), g)$. If GDP chooses a method m relevant to g with subgoals $g_1, g_2, \ldots g_l$, then there must exist a sequence of plans $\pi_1, \pi_2, \ldots, \pi_{l+1}$ that constitute π and GDP will return each π_i as a solution each goal g_i from the state $\gamma(s_0, (\pi_1 \circ \pi_2 \circ \cdots \circ \pi_{i-1}))$. Then the theorem follows. □

4.2 Domain-Independent Heuristics

GDP can easily be modified to incorporate heuristic functions similar to those used in classical planning. The modified algorithm, which we will call GDP-h (where h is the heuristic function) in the experiments, is like Algorithm 1, except that Lines 9–13 are replaced with the following:

```
sort $U$ with $h(u), \forall u \in U$
foreach $u \in U$ do
    if $u$ is an action then
        append $u$ to $\pi$; remove $g$ from $G$; $s \leftarrow \gamma(s, u)$
    else push sub($u$) into $G$
    $\pi \leftarrow \mathrm{GDP}(D, s, G, \pi)$
    if $\pi \neq$ failure then return $\pi$
return failure
```

Intuitively, this replaces the nondeterministic choice in GDP with a deterministic choice dictated by h. GDP-h uses h to order U, then attempts to decompose the current goal g in that order.

As an example, here is how we compute a variation of the Relaxed Graphplan heuristic used by the FF planner [16]. At the start of the planning process, we generate a relaxed planning graph PG from the start state s_0 to its fixpoint. Let $l_{PG}(p)$ be the first propositional level in which p appears in PG. Then $h_{s,G}(u)$, the heuristic value of applying action/method u in a state s to achieve the goals in the list G, is as follows:

$$h_{s,G}(u) =$$

$$\begin{cases} 1 + \max\limits_{p \in G} l_{PG}(p) - \max\limits_{p \in \gamma(s,u)} l_{PG}(p), & \text{if } u \text{ is an action,} \\ \max\limits_{p \in G \cup sub(u)} l_{PG}(p) - \max\limits_{p \in s} l_{PG}(p), & \text{if } u \text{ is a method.} \end{cases}$$

Intuitively, what h estimates is the distance between the first level in which the literals in G are asserted and the first level in which the current state is asserted. When u is a method, since any plan generated via u has to achieve $sub(u)$ enroute, it considers the set $G \cup sub(u)$ instead as the goal.

Note that this gives weaker heuristic values than the original FF heuristic since we do not generate a relaxed plan and use its length as the heuristic value. However, we use this variant of the heuristic since it is much more efficiently computable without compromising too much on search control. The strength of the heuristic is not as critical here as in classical planning, since the HGN methods themselves constrain what part of the space gets searched.

Method for using truck ?t to move crate ?o
    from location ?l1 to location ?l2 in city ?c:

Head: (move-within-city ?o ?t ?l1 ?l2 ?c)
Pre: ((obj-at ?o ?l1) (in-city ?l1 ?c)
    (in-city ?l2 ?c) (truck ?t ?c) (truck-at ?t ?l3))
Sub: ((truck-at ?t ?l1) (in-truck ?o ?t)
    (truck-at ?t ?l2) (obj-at ?o ?l2)))

Method for using airplane ?plane to move crate ?o
    from airport ?a1 to airport ?a2:

Head: (move-between-airports ?o ?plane ?a1 ?a2)
Pre: ((obj-at ?o ?a1) (airport ?a1)
    (airport ?a2) (airplane ?plane))
Sub: ((airplane-at ?plane ?a1) (in-airplane ?o ?plane)
    (airplane-at ?plane ?a2) (obj-at ?o ?a2)))

Method for moving ?o from location ?l1 in city ?c1
    to location ?l2 in city ?c2, via airports ?a1 and ?a2:

Head: (move-between-cities ?o ?l1 ?c1 ?l2 ?c2 ?a1 ?a2)
Pre: ((obj-at ?o ?l1) (in-city ?l1 ?c1) (in-city ?l2 ?c2)
    (different ?c1 ?c2) (airport ?a1) (airport ?a2)
    (in-city ?a1 ?c1) (in-city ?a2 ?c2))
Sub: ((obj-at ?o ?a1) (obj-at ?o ?a2) (obj-at ?o ?l2)))

Figure 4: HGN methods for transporting a package to its goal location in the Logistics domain.

5. EXPERIMENTAL EVALUATION

We implemented GDP in Common Lisp, and compared it with SHOP2 and the classical planner FF in five different planning domains:[4] These included the well-known Logistics [31], Blocks-World [2], Depots [11], and Towers of Hanoi [1] domains, and a new *3-City Routing* domain that we wrote in order to provide a domain in which the planners' domain models would not be of much help. The following questions motivated our experiments:

- *How does GDP's performance (plan quality and running time) compare with SHOP2's?* In order to investigate this question, we were careful to use domain models for SHOP2 and GDP that encoded basically the same control information.[5]
- *What is the relative difficulty of writing domain models for GDP and SHOP2?* We had no good way to measure this directly;[6] but as a proxy for it, we (i) measured the relative sizes of the SHOP2 and GDP domain models, and (ii) examined the domain models to find out the reasons for the difference in size.
- *How useful is GDP-h's heuristic function when the domain model is strong?* For this, we compared GDP-h with GDP on the Logistics, Blocks World, and Depots domains.
- *When the domain model is weak, how much help does GDP-h's heuristic function provide?* For this, we compared GDP-h's performance with GDP's on the 3-City Routing domain.
- *Since GDP-h's heuristic function is loosely based on FF's, how does GDP-h's performance compare to FF's?* For this purpose, we included FF in our experiments.
- *Is GDP as sensitive as SHOP2 is to the order in which the methods appear in the domain model?* To investigate this question, we took our domain models for SHOP2 and GDP, and rearranged the methods into a random order. In experimental results that follow, we use the names SHOP2-r and GDP-r to refer to SHOP2 and GDP with those domain models.

The GDP source code, and the HGN and HTN domain models used in our experiments, are available at http://www.cs.umd.edu/projects/planning/data/shivashankar12hierarchical/.

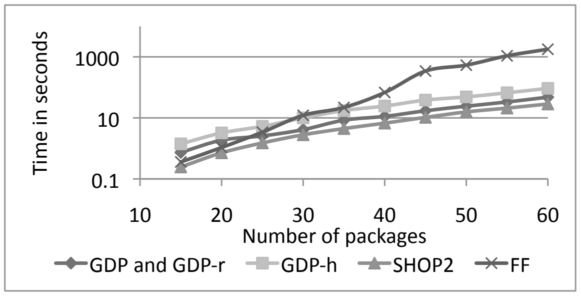
5.1 Planning Performance

To compile and execute GDP, GDP-h, and SHOP2, we used Allegro Common Lisp 8.0. For FF, we used the open-source C implementation from the FF web site. All experiments were run on 2GHz dual-core machines with 4GB RAM. We set a time limit of

[4]We used SHOP2 instead of SHOP for two reasons: (1) its algorithm is identical to SHOP's when restricted to totally-ordered subtasks, and (2) since its implementation includes many enhancements and optimizations not present in SHOP, it provides a more rigorous test of GDP.

[5]An important aspect of SHOP2's domain models is the use of Horn-clause inference to infer some of the preconditions. So that we could write GDP domain models equivalent to SHOP2's, we included an identical Horn-clause inference engine in GDP.

[6]That would have required a controlled experiment on a large number of human subjects, each of whom has equal amounts of training and experience with both GDP and SHOP2. We have no feasible way to perform such an experiment.



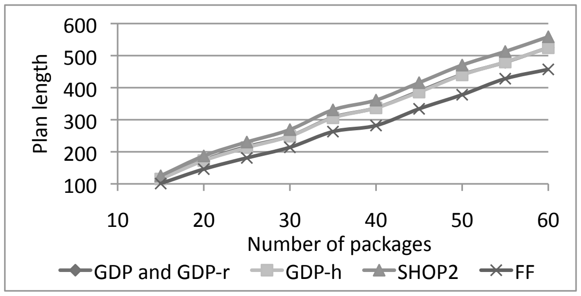
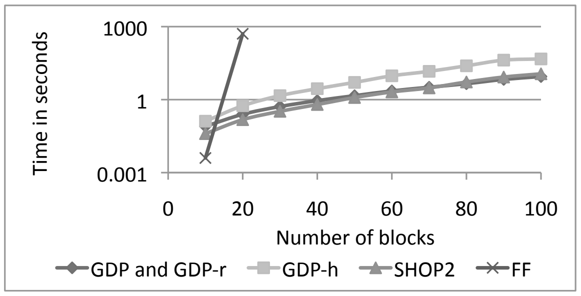


Figure 1: Average running times (in logscale) and plan lengths in the Logistics domain, as a function of the number of packages. Each data point is an average of the 10 problems from the SHOP2 distribution. There are no data points for SHOP2-r because it could not solve any of the problems. GDP and GDP-r performed identically because the methods had mutually exclusive preconditions.



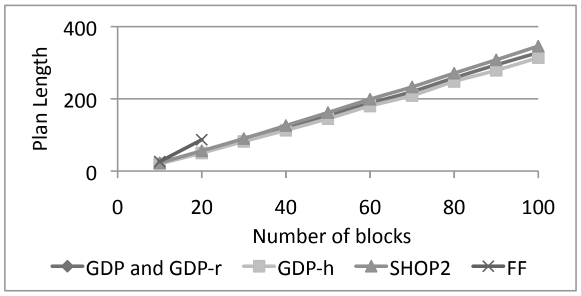


Figure 2: Average running times (in logscale) and plan lengths in the Blocks World domain, as a function of the number of blocks. Each data point is an average of 25 randomly generated problems. There are no data points for SHOP2-r because it could not solve any of the problems. GDP and GDP-r performed identically because the preconditions of the methods were mutually exclusive. FF was unable to solve problems involving more than 20 blocks.

two hours per problem, and data points not solved within the required time limit were discarded.

The Logistics Domain. For SHOP2, we used the Logistics domain model in the SHOP2 distribution. For GDP and GDP-h, we wrote the methods in Fig. 4 (these methods are easy to prove complete [28]). For the experiments, we used the Logistics Domain problems in the SHOP2 distribution. These included ten n-package problems for each of $n = 15, 20, 25, \ldots, 60$.

Figure 1 shows a comparison of running times and plan lengths of the planners in this domain. The running times of GDP, GDP-h and SHOP2 were very similar, showing that even on easy domains with strong domain models, the heuristic does not add much overhead to GDP-h's running time. FF's running times, however, grew much faster: with 60 packages, FF was nearly two orders of magnitude slower than SHOP2.

The plans produced by GDP and GDP-h were of nearly the same length, and the plans produced by SHOP2 were slightly longer. FF produced the shortest plans; this indicates that its heuristic function was slightly stronger than the relaxed version we used in GDP-h.

SHOP2-r did not terminate on any of the instances, while GDP-r performed identically to GDP. In fact, we observed that the same was true across all of the domains in our experimental study. We defer the explanation of this to Section 5.3.

The Blocks World. For SHOP2, we used the domain model included in SHOP2's distribution. For GDP and GDP-h we used a much more compact domain model consisting of three methods (shown here as pseudocode):

- **To achieve** on(x, y)
  precond: y is in its final position[7]
  subgoals: achieve `clear`(x), `clear`(y) and `on`(x, y)
- **To achieve** `clear`(x)
  precond: `on`(y, x)
  subgoals: achieve `clear`(y) and then `clear`(x)
- **To achieve** `on-table`(x)
  precond: None
  subgoals: achieve `clear`(x) and then `on-table`(x)

As shown in Figure 2, GDP and SHOP2 took nearly identical times to solve the problems, with GDP-h taking slightly longer due to its heuristic computation overhead. FF, which is known to have problems with the Blocks World [2], was unable to solve problems with more than 20 blocks.

As shown in the figure, GDP, GDP-h and SHOP2 produced solution plans of similar length, with GDP-h producing the shortest plans. FF produced significantly longer plans than the other three planners, even for the problems it managed to solve.

The Depots Domain. For SHOP2, we used the Depots domain model from the SHOP2 distribution. For GDP and GDP-h, we simply stitched together relevant parts of the Logistics and Blocks-World domain models, and adapted them to obtain an HGN Depots domain model that encoded the same control information.

As shown in Figure 3, GDP and SHOP2 took similar times to solve the problems. However, GDP-h's running times grew much faster than GDP or SHOP2, indicating that the overhead of the heuristic can increase with the complexity of the domain. FF was unable to solve any problems of size greater than 24 crates.

[7]Inferred using Horn clauses (see footnote 5).

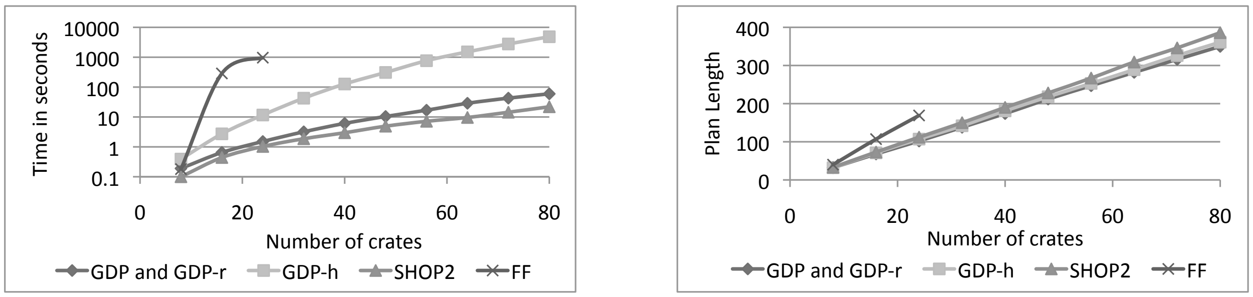


Figure 3: Average running times (in logscale) and plan lengths in the Depots domain, as a function of the number of crates. Each data point is an average of 25 randomly generated problems. There are no data points for SHOP2-r because it could not solve any of the problems. GDP and GDP-r performed identically because the preconditions of the methods were mutually exclusive. FF was unable to solve problems involving more than 24 crates.

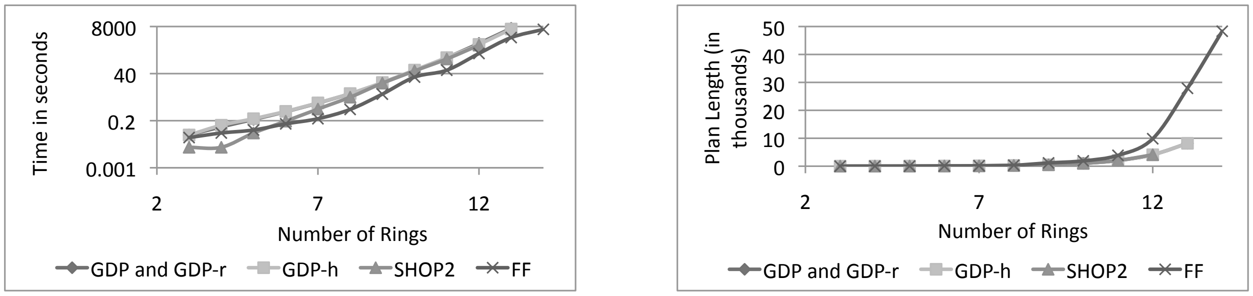


Figure 5: Average running times (in logscale) and plan lengths in the Towers of Hanoi domain, as a function of the number of rings. Each data point is an average of 10 runs. There are no data points for SHOP2-r because it could not solve any of the problems. GDP and GDP-r performed identically because the preconditions of the methods were mutually exclusive.

With respect to plan lengths, GDP and GDP-h produced almost identical plans, with SHOP2 producing slightly longer plans than GDP. For the problem sizes it could handle, FF produced significantly longer plans than the other three planners.

Towers of Hanoi. We wrote domain models for SHOP2 and GDP that encoded an algorithm to produce optimal solution plans (i.e., length $2^n - 1$ for an n-ring problem).

Figure 5 shows the planners' runtimes and plan lengths. As expected, GDP, GDP-h and SHOP2 returned optimal plans whereas FF returned significantly sub-optimal plans.

However, while GDP, GDP-h, SHOP2 and FF had similar runtimes up to problems of size 12, SHOP2 could not solve the larger problems due to a stack overflow, and GDP could not solve the 14-ring problem within the time limit. We believe this is basically an implementation issue: both GDP and SHOP2 had recursion stacks of exponential size, whereas FF (since it never backtracks) did not.

3-City Routing. In the four planning domains discussed above, the GDP and SHOP2 domain models pruned the search space enough that GDP-h's heuristic function could not reduce it much further (if at all). In order to examine the performance of the planners in a domain with a weak domain model, we constructed the *3-City Routing* domain. In this domain, there are three cities c_1, c_2 and c_3, each containing n locations internally connected by a network of randomly chosen roads. In addition, there is one road between a randomly chosen location in c_1 and a randomly chosen location in c_2, and similarly another road between locations in c_2 and c_3. The problem is to get from a location in c_1 or c_3 to a goal location in c_2.

We randomly generated 25 planning problems for each value of n, with n varying from 10 to 100. For the road networks, we used near-complete graphs in which 20% of the edges were removed at random. Note that while solutions to such problems are typically very short, the search space has extremely high branching factor, i.e. of the order of n. For GDP and GDP-h, we used a single HGN method, shown here as pseudocode:

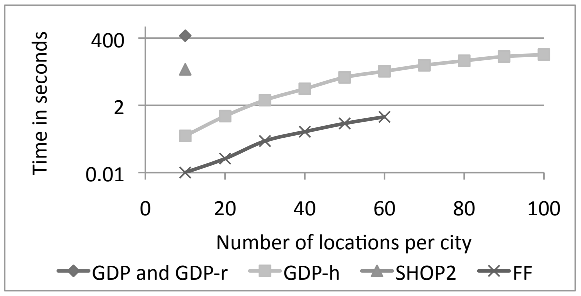
```
• To achieve at($b$)
  precond: at($a$), adjacent($c, b$)
  subgoals: achieve at($c$) and then at($b$)
```

By applying this method recursively, the planner can do a backward search recursively from the goal location to the start location.

To accomplish the same backward search in SHOP2, we needed to give it three methods, one for each of the following cases: (1) goal location same as the initial location, (2) goal location one step away from the initial location, and (3) arbitrary distance between the goal and initial locations.

As Figure 6 shows, GDP and SHOP2 did not solve the randomly generated problems except the ones of size 10, returning very poor solutions and taking large amounts of time in the process. GDP-h, on the other hand solved all the planning problems quickly, returning near-optimal solutions. The reason for the success of GDP-h is that the domain knowledge specified above induce an unguided backward search in the state space and the planner uses the domain-independent heuristic to select its path to the goal.

FF was able to solve all problems up to $n = 60$ locations, after which it could not even complete parsing the problem file. We believe this has to do with FF grounding all the actions right in the beginning, which it could not do for the larger problems.



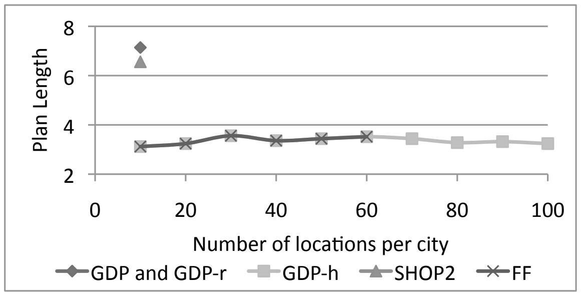


Figure 6: Average running times (in logscale) and plan lengths in the 3-City Routing domain, as a function of the number of locations per city. Each data point is an average of 25 randomly generated problems. There are no data points for SHOP2-r because it couldn't solve any problems. FF couldn't solve problems involving more than 60 locations while GDP and SHOP2 could not solve problems with more than 10 locations. GDP and GDP-r performed identically because there was only one method in the domain model.

5.2 Domain Authoring

When writing the domain models for our experiments, it seemed to us that writing the GDP domain models was easier than writing the SHOP2 domain models—so we made measurements to try to verify whether this subjective impression was correct.

Figure 7 compares the sizes of the HGN and HTN domain descriptions of the planning domains. In almost all of them, the domain models for GDP were much smaller than those for SHOP2. There are three main reasons why:

- To specify how to achieve a logical formula p in the HTN formalism, one must create a new task name t and one or more methods such that (i) the plans generated by these methods will make p true and (ii) the methods have syntactic tags saying that they are relevant for accomplishing t. If there is another method m' that makes p true but does not have such a syntactic tag, the planner will never consider using m' when it is trying to achieve p. In contrast, relevance of a method in HGN planning is similar to relevance of an action in classical planning: if the effects of m' include p, then m' is relevant for p.
- Furthermore, suppose p is a conjunct $p = p_1 \wedge \ldots \wedge p_k$ and there are methods $m_1, \ldots, m_k$ that can achieve $p_1, \ldots, p_k$ piecemeal. In HGN planning, each of these methods is relevant for p if it achieves some part of p and does not negate any other part of p. In contrast, those methods are not relevant for p in HTN planning unless the domain description includes (i) a method that decomposes t into tasks corresponding to subsets of $p_1, \ldots, p_k$, (ii) methods for those tasks, and (iii) an explicit check for deleted-condition interactions.[8] This can cause the number of HTN methods to be much larger (in some cases exponentially larger) than the number of HGN methods.
- In recursive HTN methods, a "base-case method" is needed for the case where nothing needs to be done. In recursive HGN methods, no such method is needed, because the semantics of goal achievement already provide that if a goal is already true, nothing needs to be done.

The Towers of Hanoi domain was the only one where the HGN domain model was larger than the corresponding HTN domain model. In this domain, the HGN domain model needed two extra actions, *enable* and *disable*, to alternately insert and delete a special atom in the state. They were needed in order to control the applicability of the *move* operator to ensure optimality.

[8]In the HTN formalism in [7], one way to accomplish (iii) is to specify t as the syntactic form $achieve(p)$, which adds a constraint that p must be true after achieving t. But that approach is inefficient in practice because it can cause lots of backtracking. In the blocks-world implementation in the SHOP2 distribution, (iii) is accomplished without backtracking by using Horn-clause inference to do some elaborate reasoning about stacks of blocks.

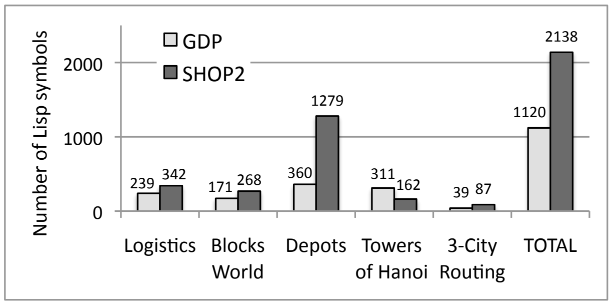


Figure 7: Sizes (number of Lisp symbols) of the GDP and SHOP2 domain models.

5.3 Discussion

We have seen from our experimental study that HGN domain models are considerably more succinct than the corresponding HTN models. We also saw that this compactness came at no extra cost; GDP's performance compared favorably to that of SHOP2's across all domains. Runtimes of the heuristic-enhanced planner GDP-h were, for the most part, comparable to those of GDP's and SHOP2's, indicating that our heuristic does not add a significant overhead to the planning time. Lengths of plans returned by GDP-h were nearly always better than GDP's and SHOP2's. This difference was especially amplified in cases where the planners had weak domain models; in such cases, the heuristic provided critical search control to GDP-h, thus helping it terminate quickly with good solutions.

In our experiments, SHOP2-r did not solve any of the problems. The reason for this was SHOP2's heavy reliance on the method order in its domain model, especially the placement of "base-cases" for recursion. For GDP-r, shuffling HGN methods had no effect at all on performance. This was because the methods in our HGN domain models had mutually exclusive preconditions, hence at most one of them was applicable. In domains where more than one method is applicable at once, GDP-r should (like SHOP2-r) perform badly when presented with methods in the wrong order.

6. CONCLUSIONS

Our original motivation for HGN planning was to provide a task semantics that corresponded readily to the goal semantics of classical planning and gave stronger soundness guarantees when applied to classical planning domains. But our work also produced two other benefits that we had not originally expected: writing HGN methods was usually much simpler than writing HTN methods, and the HGN formalism can easily incorporate HGN extensions of classical-style heuristic functions to guide the search.

Our proof that HGN planning is as expressive as totally-ordered HTN planning means that it is capable of encoding complicated control knowledge, one of the main strengths of HTN planning. This suggests that HGN planning has the potential to be very useful both for research purposes and in practical applications.

With that in mind, we have several ideas for future work:

- GDP currently supports only totally ordered subtasks. We intend to generalize HGNs to allow partially-ordered subtasks.
- We intend to generalize HGNs to allow *partial* sets of methods analogous to the ones in [1]. This will provide an interesting hybrid of task decomposition and classical planning. Furthermore, it will make writing HGN domain models even easier while preserving the efficiency advantages of HGN planning.
- Replanning in dynamic environments is becoming an increasingly important research topic. We believe HGN planning is a promising approach for this topic.
- HTN planning has been extended to accommodate actions with nondeterministic outcomes [18], temporal planning [4, 15], and to consult external information sources [19]. It should be straightforward to make similar extensions to HGN planning.

Acknowledgments. This work was supported, in part, by DARPA and the U.S. Army Research Laboratory under contract W911NF-11-C-0037, and by a UMIACS New Research Frontiers Award. The views expressed are those of the authors and do not reflect the official policy or position of the funders.

7. REFERENCES

[1] R. Alford, U. Kuter, and D. S. Nau. Translating HTNs to PDDL: A small amount of domain knowledge can go a long way. In *IJCAI*, July 2009.

[2] F. Bacchus. The AIPS '00 planning competition. *AI Mag.*, 22(1):47–56, 2001.

[3] B. Bonet and H. Geffner. Planning as heuristic search: New results. In *ECP*, Durham, UK, 1999.

[4] L. Castillo, J. Fdez-Olivares, O. Garcıa-Pérez, and F. Palao. Efficiently handling temporal knowledge in an HTN planner. In *ICAPS*, 2006.

[5] M. Chung, M. Buro, and J. Schaeffer. Monte carlo planning in RTS games. In *IEEE Symp. Comp. Intel. Games*, 2005.

[6] K. Currie and A. Tate. O-Plan: The open planning architecture. *Artif. Intell.*, 52(1):49–86, 1991.

[7] K. Erol, J. Hendler, and D. S. Nau. HTN planning: Complexity and expressivity. In *AAAI*, 1994.

[8] K. Erol, J. Hendler, and D. S. Nau. UMCP: A sound and complete procedure for hierarchical task-network planning. In *AIPS*, pages 249–254, June 1994. ICAPS 2009 influential paper honorable mention.

[9] K. Erol, J. Hendler, and D. S. Nau. Complexity results for hierarchical task-network planning. *AMAI*, 18:69–93, 1996.

[10] T. A. Estlin, S. Chien, and X. Wang. An argument for a hybrid HTN/operator-based approach to planning. In *ECP*, pages 184–196, 1997.

[11] M. Fox and D. Long. International planning competition, 2002. http://planning.cis.strath.ac.uk/competition.

[12] M. P. Georgeff and A. L. Lansky. Reactive reasoning and planning. In *AAAI*, pages 677–682, 1987.

[13] A. Gerevini, U. Kuter, D. S. Nau, A. Saetti, and N. Waisbrot. Combining domain-independent planning and HTN planning. In *ECAI*, pages 573–577, July 2008.

[14] M. Ghallab, D. S. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. May 2004.

[15] R. Goldman. Durative planning in HTNs. In *ICAPS*, 2006.

[16] J. Hoffmann and B. Nebel. The FF planning system. *JAIR*, 14:253–302, 2001.

[17] S. Kambhampati, A. Mali, and B. Srivastava. Hybrid planning for partially hierarchical domains. In *AAAI*, pages 882–888, 1998.

[18] U. Kuter, D. S. Nau, M. Pistore, and P. Traverso. Task decomposition on abstract states, for planning under nondeterminism. *Artif. Intell.*, 173:669–695, 2009.

[19] U. Kuter, E. Sirin, D. S. Nau, B. Parsia, and J. Hendler. Information gathering during planning for web service composition. *JWS*, 3(2-3):183–205, 2005.

[20] B. Marthi, S. Russell, and J. Wolfe. Angelic semantics for high-level actions. In *ICAPS*, 2007.

[21] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, H. Muñoz-Avila, J. W. Murdock, D. Wu, and F. Yaman. Applications of SHOP and SHOP2. *IEEE Intell. Syst.*, 20(2):34–41, Mar.-Apr. 2005.

[22] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman. SHOP2: An HTN planning system. *JAIR*, 20:379–404, Dec. 2003.

[23] D. S. Nau, Y. Cao, A. Lotem, and H. Muñoz-Avila. SHOP: Simple hierarchical ordered planner. In T. Dean, editor, *IJCAI*, pages 968–973, Aug. 1999.

[24] D. S. Nau, Y. Cao, A. Lotem, and H. Muñoz-Avila. The SHOP planning system. *AI Mag.*, 2001.

[25] F. Sailer, M. Buro, and M. Lanctot. Adversarial planning through strategy simulation. In *IEEE Symp. Comp. Intel. Games*, 2007.

[26] B. Schattenberg. *Hybrid Planning & Scheduling*. PhD thesis, Universität Ulm, Mar. 2009.

[27] R. Sherwood, A. Mishkin, T. Estlin, S. Chien, P. Backes, B. Cooper, S. Maxwell, and G. Rabideau. Autonomously generating operations sequences for a mars rover using artifical intelligence-based planning. In *IROS*, Oct. 2001.

[28] V. Shivashankar, U. Kuter, and D. Nau. Hierarchical goal network planning: Initial results. Technical Report CS-TR-4983, Univ. of Maryland, May 2011.

[29] S. J. J. Smith, D. S. Nau, and T. Throop. Computer bridge: A big win for AI planning. *AI Magazine*, 19(2):93–105, 1998.

[30] A. Tate, B. Drabble, and R. Kirby. *O-Plan2: An Architecture for Command, Planning and Control*. 1994.

[31] M. M. Veloso. Learning by analogical reasoning in general problem solving. PhD thesis CMU-CS-92-174, Carnegie Mellon University, 1992.

[32] D. E. Wilkins. Domain-independent planning: Representation and plan generation. *Artif. Intell.*, 22(3):269–301, Apr. 1984.

[33] D. E. Wilkins. *Practical Planning: Extending the Classical AI Planning Paradigm*. San Mateo, CA, 1988.

DISCOVERHISTORY: Understanding the Past in Planning and Execution

Matthew Molineaux
Knexus Research Corporation
9120 Beachway Lane
Springfield, VA 22153 USA
matthew.molineaux
@knexusresearch.com

Ugur Kuter
Smart Information Flow Technologies
211 North 1st Street
Minneapolis, MN 55401 USA
ukuter@sift.net

Matthew Klenk
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304 USA
Matthew.Klenk@parc.com

ABSTRACT

We consider the problem of automated planning and control for an execution agent operating in environments that are partially-observable with deterministic exogenous events. We describe a new formalism and a new algorithm, DISCOVERHISTORY, that enables our agent, DHAgent, to proactively expand its knowledge of the environment during execution by forming explanations that reveal information about the world. We describe how DHAgent uses this information to improve the projections made during planning. Finally, we present an ablation study that examines the impact of explanation generation on execution performance. The results of this study demonstrate that our approach significantly increases the goal achievement success rate of DHAgent against an ablated version that does not perform explanation.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Plan execution, formation, and generation; I.2.3 [**Deduction and Theorem Proving**]: Abductive Reasoning; I.2.11 [**Distributed Artificial Intelligence**]: Intelligent Agents

General Terms

Algorithms

Keywords

Planning, execution, abductive reasoning, explanation generation

1. INTRODUCTION

In real-world tasks, perceptions are incomplete and the world is constantly changing due to exogenous events. Such events often cause plans to fail, and understanding why they occur is sometimes necessary to achieve strong performance. Real-world agents do not necessarily observe such events directly, so they must reason that changes in the world are explained by events.

Consider the following real-world example of the power of explanation. In May 2005, NASA's Opportunity rover was crossing a dune on the surface of Mars when its human operators noticed an *inconsistency* with their expectations: Opportunity was not moving as much as expected [23]. The operators were not able to observe the surroundings of the rover fully and precisely, but they nevertheless *explained* this inconsistency by *assuming* that the rover was stuck in loose soil. This explanation enabled the operators to formulate a new plan to escape from the unobserved loose soil and continue the mission.

The focus of this work is on algorthmically explaining the history of a partially-observable, dynamic environment in order to understand prediction failures and thereby improve future predictions. To accomplish this, we have devised an algorithm that models change in terms of deterministic exogenous events that can be predicted and reasoned about. This reasoning about prediction failures contrasts with typical work on replanning approaches, which focuses on resolving failures between individual causal links in the plan, and does not attempt to understand the causes of that failure (e.g., [18, 10, 24, 1]). In the diagnosis field, work on constructing plan diagnoses addresses similar issues of constructing histories during execution in planning domains (e.g., [6], [4]) and discrete-event systems (e.g., [20]), but this work is based on a traditional representation of exogenous events as *actions* conducted by another agent or nature. Our work represents events as natural consequences that occur automatically, rather than by choice, which is not supported by current replanning or diagnosis systems. Finally, the SDR system (e.g., [19]) uses regression to generate causal explanations of an agent's history based on a contingent planning domain. In this work, different possible outcomes of actions are possible, which is modeled using conditional effects. While this work is similar in aim to ours, the representation understood by the SDR system is strictly less expressive than the representation used in our work.

Our contributions are the following:

- We describe a formalism for reasoning about the causes of inconsistencies between observations and expectations that arise during plan execution. This formalism includes a novel model for deterministic exogenous events and their effects on the world that models partial observability using a distinction between observable and hidden facts.
- We describe DISCOVERHISTORY, an algorithm that generates abductive explanations of possible histories during plan execution.
- We describe DHAgent, a simple planning and execution agent that uses DISCOVERHISTORY to maintain its description of the world and reason about hidden state. This allows DHAgent to (1) discard plans which would fail due to the occurrence of predicted events and (2) take advantage of opportunities afforded by future events.
- We discuss experiments in two planning domains, modified versions of the Rovers and Satellite domains from past International

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

Planning Competitions, to show that DHAgent's performance is improved by a statistically significant margin when using DISCOVERHISTORY for explanation. These experiments do not compare DISCOVERHISTORY with existing algorithms, which, as we describe later, do not support these domains.

2. DEFINITIONS AND NOTATION

In this section, we describe a new formalism for explanation generation. Representations and reasoning in this formalism use observations, actions, and events as the basic building blocks, so that an agent can conduct planning and explanation using a single domain model. This reduces the knowledge engineering burden by making an additional use of existing knowledge; it works because explanation is an inversion of planning.

Unlike typical formalisms from planning and diagnosis, ours models events as deterministic. These are somewhat similar to conditional effects on actions, since they occur deterministically following the execution of an action. However, deterministic events can be triggered by *any* action, or another event; modeling the same information as effects of actions would require the domain author to consider the effects of all series of events that could ever happen following an action's execution. Therefore, the size of an equivalent domain model using conditional effects rather than events would be exponential in the number of events to be represented.

The advantages of representing deterministic exogenous events are twofold: (1) we can determine (predictively or after the fact) the exact time when events must occur, reducing the set of potential explanations for a given series of observations and (2) interacting effects can combine without causing an explosion in the number of actions or events considered. Unpredictability in environments under our representation arises only from hidden facts, not a "choice" made by an environment as to whether an event will occur.

2.1 Basics

We use the standard definitions from classical planning for variable and constant symbols, logical predicates and atoms, literals, groundings of literals, propositions, planning operators and actions [3, Chapter 2].

Let $\mathcal{P}$ be the finite set of all propositions describing a planning environment; the *state* of the environment is described by assigning a value to each proposition in P. A planning environment is partially observable if an agent only has access to the environment through *observations* which do not cover the complete state. We let $\mathcal{P}_{obs}$ be the set of all propositions that the agent will observe when true. An observation associates a truth value with each of these propositions. Let $\mathcal{P}_{hidden}$ be a set of *hidden* propositions representing aspects of the world an agent cannot observe; for example, the exact location may be hidden to a robot with no GPS contact.

An *event template* is defined syntactically the same as a classical planning operator: $(\mathsf{name}, \mathsf{preconds}, \mathsf{effects})$, where name, the *name* of the event, preconds and effects, the *preconditions* and *effects* of the event, are sets of literals. We use $\mathsf{effects}^-$ and $\mathsf{effects}^+$ to denote the negative and positive literals in effects, respectively. An *event* is a ground instance of an event template. We assume that an event always occurs immediately when all of its preconditions are met in the world. After each action, any events triggered by that action occur, followed by events triggered by those events, etc. When no more events occur, the agent receives a new observation.

2.2 Explanations

We formalize the planning agent's knowledge about the changes in its environment as an *explanation* of the world. We define a finite set of symbols $\mathcal{T} = \{t_0, t_1, t_2, \ldots, t_n\}$, called *occurrence points*. An *ordering relation* between two occurrence points is denoted as $t_i \prec t_j$, where $t_i, t_j \in \mathcal{T}$.

There are three types of *occurrences*. An *observation occurrence* is a pair of the form (obs, t) where obs is an observation. An *action occurrence* is a pair of the form (a, t) where a is an action. Finally, an *event occurrence* is a pair (e, t) where e is an event. In all of the occurrence forms, t is an occurrence point. Given an occurrence o, we define occ as a function such that $\mathsf{occ}(o) \mapsto t$; that is, occ refers to the occurrence point t of any observation, action, or event.

An *execution history* is a finite sequence of observations and actions $\mathsf{obs}_0, a_1, \mathsf{obs}_1, a_2, \ldots, a_k, \mathsf{obs}_{k+1}$. A planning agent's *explanation* of the world given an execution history is a tuple $\chi = (C, R)$ such that C is a finite set of occurrences that includes each obs_i for $i = 0, \ldots, k-1$ and each action a_j for $j = 1, \ldots, k$ for some number k. C may also include zero or more event occurrences that happened according to that explanation. R is a partial ordering over a subset of C, described by ordering relations $\mathsf{occ}(o_i) \prec \mathsf{occ}(o_j)$ such that $o_i, o_j \in C$. As a shorthand, we sometimes will say $o_i \prec o_j$ if and only if $\mathsf{occ}(o_i) \prec \mathsf{occ}(o_j)$.

We use the definitions $\mathsf{knownbefore}(p, o)$ and $\mathsf{knownafter}(p, o)$ to refer to the value of a proposition p before or after an occurrence $o \in C$ occurs. Let o be an action or event occurrence. Then, the relation $\mathsf{knownbefore}(p, o)$ is true iff $p \in \mathsf{preconds}(o)$. Similarly, the relation $\mathsf{knownafter}(p, o)$ is true iff $p \in \mathsf{effects}(o)$. If o is an observation occurrence and $p \in \mathsf{obs}$, then both $\mathsf{knownbefore}(p, o)$ and $\mathsf{knownafter}(p, o)$ are true, and otherwise are false.

We say that an occurrence o is *relevant* to a proposition p if the following holds:

$$\begin{aligned}\mathsf{relevant}(p, o) \equiv\ & \mathsf{knownafter}(p, o) \lor \mathsf{knownafter}(\neg p, o) \lor \\ & \mathsf{knownbefore}(p, o) \lor \mathsf{knownbefore}(\neg p, o).\end{aligned}$$

We use the predicates $\mathsf{prior}(o, p)$ and $\mathsf{next}(o, p)$ to refer to the prior and next occurrence relevant to a proposition p. That is to say, $\mathsf{prior}(o, p) = \{o' \mid \mathsf{relevant}(p, o') \land \neg\exists o'' s.t. \mathsf{relevant}(p, o'') \land o' \prec o'' \prec o\}$. Similarly, $\mathsf{next}(o, p) = \{o' \mid \mathsf{relevant}(p, o') \land \neg\exists o'' s.t. \mathsf{relevant}(p, o'') \land o \prec o'' \prec o'\}$.

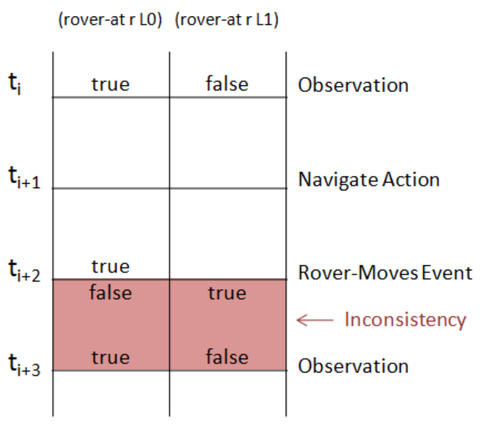
2.3 Plausibility

The *proximate cause* of an event occurrence (e, t) is an occurrence o that satisfies the following three conditions with respect to some proposition p: (1) $p \in \mathsf{preconds}(e)$, (2) $\mathsf{knownafter}(p, o)$, and (3) there is no other occurrence o' such that $o \prec o' \prec (e, t)$. Every event occurrence (e, t), must have at least one proximate cause, so by condition 3, every event occurrence must occur immediately after its preconditions are satisfied.

An *inconsistency* is a tuple (p, o, o') where o and o' are two occurrences in χ such that $\mathsf{knownafter}(\neg p, o)$, $\mathsf{knownbefore}(p, o')$, and there is no other occurrence o'' such that $o \prec o'' \prec o' \in R$ and p is relevant to o''.

An explanation $\chi = (C, R)$ is *plausible* if and only if the following holds:

1. There are no inconsistencies in χ;
2. Every event occurrence $(e, t) \in \chi$ has a proximate cause in χ;
3. For every pair of simultaneous occurrences such that $o, o' \in C$ and $occ(o) = occ(o')$, there may be no conflicts before or after: for all p, $\mathsf{knownafter}(p, o) \implies \neg\mathsf{knownafter}(\neg p, o')$, and $\mathsf{knownbefore}(p, o) \implies \neg\mathsf{knownbefore}(\neg p, o')$.
4. If $\mathsf{preconds}(e)$ of an event e are all satisfied at an occurrence point t, e is in χ at t;



```
(:action navigate
  :parameters (?r - rover ?dir - dir)
  :precondition
    (and (available ?r)  (>= (energy ?r) 8))
  :effect
    (and (attempting-move ?r ?dir) (decrease (energy ?r) 8)))

(:event rover-moves
  :parameters (?r - rover)
  :precondition
    (and (attempting-move ?r ?dir)  (not (blocked ?r))
         (rover-at ?r ?src) (can_traverse ?r ?src ?dest)
         (visible ?src ?dest))
   :effect
    (and (not (rover-at ?r ?src)) (at ?r ?dest)
       (not (attempting-move ?r ?dir))))
```

Figure 1: Example of an inconsistent explanation, with occurrence points ordered on the left hand side. Relevant action and event descriptions are given on the right. Boolean values are the knownbefore and knownafter relations; for example, the value "false" at top right indicates that the relation knownbefore(($\neg$ (rover-at r L1)), o_i) holds.

Algorithm 1: A high-level description of DISCOVERHISTORY.

```
1  Procedure DiscoverHistory ($\chi$)
2  begin
3      if Inconsistencies($\chi$) = $\emptyset$ then
4          $\chi \leftarrow$ FindExtraEvents ($\chi$)
5          if $\chi = \emptyset$ then return $\emptyset$
6          if Inconsistencies($\chi$) = $\emptyset$ then return $\{\chi\}$
7      arbitrarily select an $i \in$ Inconsistencies($\chi$)
8      $X \leftarrow$ Refine ($\chi, i$)
9      foreach $\chi_{new} \in X$ do
10         $X \leftarrow X \cup$ DiscoverHistory ($\chi_{new}$)
11     return $X$
```

EXAMPLE 1. *Suppose that a rover r attempts to move after its wheel has, unobserved, become stuck. Figure 1 illustrates part of an inconsistent explanation, which includes the prior observation occurrence at t_i, in which the rover is observed at location* L0*; followed by a navigate action occurrence, illustrating the rover's attempt to move, at t_{i+1} directing the rover to location* L1*; followed by an event occurrence illustrating the predicted event o_{i+2} at t_{i+2} that changes the rover's location from* L0 *to* L1*; followed by the most recent observation occurrence o_{i+3} at t_{i+3}, which states that the rover is at* L0*.*

There are two inconsistencies in this explanation, between the event occurrence and observation occurrence: $\langle$(rover-at r L0), $o_{i+2}, o_{i+3}\rangle$ *and* $\langle\neg$(rover-at r L1), $o_{i+2}, o_{i+3}\rangle$*.*

3. GENERATING ABDUCTIVE EXPLANATIONS

This section describes DISCOVERHISTORY, our search algorithm for generating plausible explanations by recursively applying refinements to implausible explanations. DISCOVERHISTORY is designed to find possible histories of a partially-observable dynamic environment. DISCOVERHISTORY generates successive explanations by attempting to resolve inconsistencies in the current explanation given its observations.

Algorithm 1 shows a high-level description of DISCOVERHISTORY. The base case occurs when the current explanation is plausible. In the recursive case, REFINE chooses an inconsistency and generates a set of explanations that resolve it. Each such explanation is then recursively considered by DISCOVERHISTORY to remove any remaining inconsistencies.

Let χ be a planner's current explanation of the world, which includes obs, the most recent observation received by the agent. DISCOVERHISTORY starts by finding all inconsistencies in the current explanation χ by calling Inconsistencies(χ). If none are present, then the first condition for plausibility is met, so the other conditions are checked to ensure plausibility (see Section 2.3). Plausibility condition 3 is assumed to be true of the initial explanation, and are maintained throughout the resolution process by removing events that contradict any new additions. Once no inconsistencies exist, therefore, FINDEXTRAEVENTS checks plausibility conditions 2 and 4. Condition 2 requires that any explanation containing events with no proximate cause be rejected; condition 4 requires that any new events caused by the changes to χ be added to χ as well. These new events may cause new inconsistencies. Otherwise, all four conditions for a plausible explanation are met and DISCOVERHISTORY returns the explanation χ in Line 6.

In Line 7, DISCOVERHISTORY selects an inconsistency i from Inconsistencies(χ) and attempts to resolve it.[1] The REFINE subroutine (Line 8 of Algorithm 1) finds all explanations that result from resolving that inconsistency and recursively calls itself on each. We further describe REFINE below.

We refer to the initial explanation constructed by the agent as χ_0. We use this explanation as the basis for our definition of goodness. An explanation χ_a is *better than* χ_b if DISCOVERHISTORY requires fewer changes to transform χ_0 into χ_a than χ_b. We use an iterative deepening search to find all explanations at a minimum depth. Since each invocation of REFINE adds a single change to its base explanation, these explanations are the *best* explanations by our goodness definition. To prevent searches from continuing indefinitely, we employ a maximum depth bound (not shown in pseudocode). The search conducted by DISCOVERHISTORY has a

[1]Any inconsistency may be selected; it does not affect the correctness of the algorithm. It may affect the algorithm's efficiency, but we do not discuss this topic further.

Algorithm 2: Subroutines REFINE and REMOVEOCC

```
1  Procedure REFINE ($\chi = (C, R), i =$(p,o,o'))
2  begin
3      $X \leftarrow \emptyset$
       /* Adding a new event occurrence */
4      foreach $(t, t') \in \mathcal{T} \times \mathcal{T} : t' \not\prec t$ do
5          if $\mathsf{occ}(o) \prec t \prec \mathsf{occ}(o') \wedge \mathsf{occ}(o) \prec t' \prec \mathsf{occ}(o')$
           then
               /* $(t, t')$ is a possible interval in which the */
               /* occurrence $o$ could happen */
6              foreach $e \in E : \mathsf{effects}(e) \models p$ do
7                  new symbol $t''$
8                  $o'' \leftarrow (e, t'')$            // New event occurrence
9                  $\chi_{new} \leftarrow (C + o'', R + t \prec t'' \prec t')$
10                 $X \leftarrow X + \chi_{new}$      // New explanation added
       /* Removing an occurrence */
11     if $o \notin \chi_0 \wedge o$ is an event occurrence then
12         $X \leftarrow X \cup$ REMOVEOCC $(\chi, o)$
13     if $o' \notin \chi_0 \wedge o'$ is an event occurrence then
14         $X \leftarrow X \cup$ REMOVEOCC $(\chi, o')$
       /* Hypothesizing an initial value */
15     if $p \in \mathcal{P}_{hidden} \wedge o = occ_0$ then
16         $X \leftarrow X + (C + (e_p, t_0), R)$               // New initial
17     return $X$                                    // value occurrence
18
19 Procedure REMOVEOCC ($\chi = (C, R), o$)
20 begin
21     $X \leftarrow \emptyset$
22     foreach $p \in \mathsf{preconds}(o)$ do
23         $X \leftarrow X + (C \setminus o + (e_r, \mathsf{occ}(o)), R)$   // New removal
24     return $X$                                    // occurrence
```

maximum depth of $|C| * (2^{|E|})^2$ and a maximum branching factor of $|E|$, where E is the finite set of all possible event occurrences.

As shown in algorithm 2, there are three possible ways for REFINE to resolve an inconsistency in an explanation, each of which has different conditions for applicability. They are: (1) adding a new occurrence, (2) removing an occurrence, and (3) hypothesizing a different initial value for some proposition. All applicable methods must be tried, resulting in multiple explanations. Each resolution may create or resolve other inconsistencies, so the inconsistencies found in refined explanations are not necessarily a subset of those found in the parent. To save space, we omit the pseudo-code for REFINE. We detail the resolution strategies below.

3.1 The REFINE Subroutine

3.1.1 *Adding a New Event Occurrence*

Let $\chi = (C, R)$ be an explanation with an inconsistency $i = (p, o, o')$. One way to resolve an inconsistency is to show that some occurrence changed the value of a literal in between the preceding occurrence o and the following occurrence o'. This occurrence must be an event o'' relevant to p such that $o \prec o'' \prec o'$.

To find such occurrences REFINE considers every possible consecutive ordered pair of occurrence points (t, t') between $occ(o)$ and $occ(o')$, given the partial-ordering R (lines 3-4). For each such pair (t, t') and every e such that $\mathsf{effects}(e) \models p$ (line 5), REFINE creates a new occurrence point t'' (line 5) and a new event occurrence $o'' = (e, t'')$. Then the algorithm adds o'' into C and updates the partial ordering R with $t \prec t'' \prec t'$. This results in one new explanation that does not contain the inconsistency i for each event

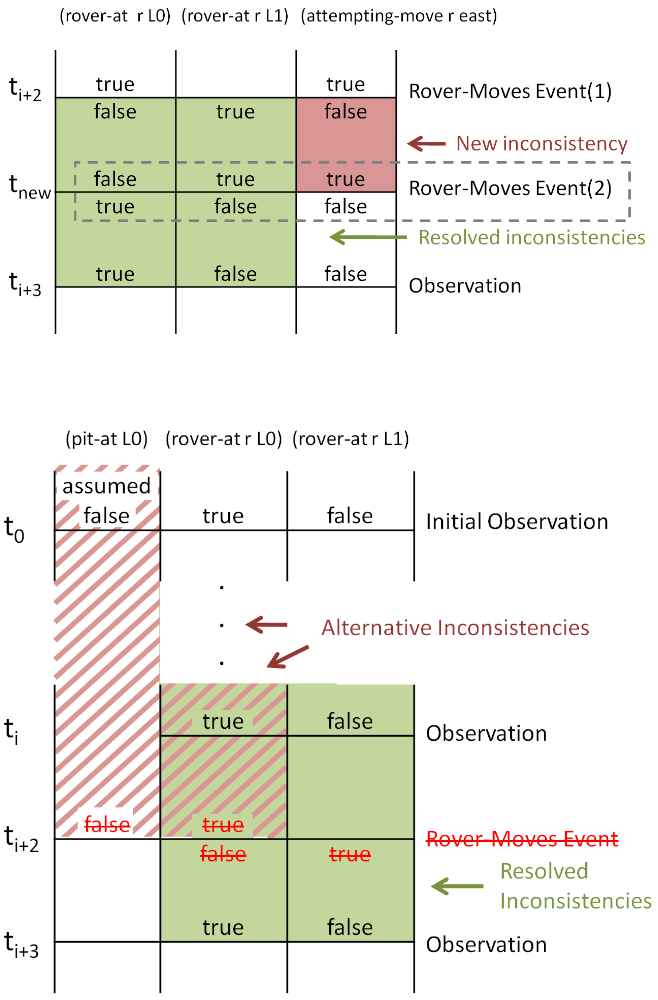


Figure 2: (a) Example of adding an occurrence (left). (b) Example of removing an occurrence (right).

$e : \mathsf{effects}(e) \models p$.

EXAMPLE 2. *Continuing Example 1, the event Rover-Moves causes the rover to enter a different location, so it could be added to resolve the inconsistency* ($\neg$ (rover-at r L1), o_{i+2}, o_{i+3}). *In order for this to work, a new occurrence must be added between t_{i+2} and t_{i+3} (see Figure 2(a)).* REFINE *creates a new explanation with added occurrence $o_{new} = (e, t_{new})$ and new ordering relations $t_{i+2} \prec t_{new} \prec t_{i+3}$. A new inconsistency is also generated (see Figure 2(a)). The new inconsistency occurs because another precondition of the Rover-Moves event is the literal* (attempting-move r east), *which is false after such an event occurs. The new inconsistency, (*(attempting-move r east), o_{i+2}, o_{new}*), will need to be eliminated in a recursive call to* DISCOVERHISTORY.

3.1.2 *Removing an occurrence*

Another possible way to resolve (p, o, o'), where o and/or o' is an event, is to refine the current explanation to generate new explanations in which either o or o' is removed.

REFINE checks both o and o' for presence in χ_0; if either was, then removing it would cause a cycle. Otherwise, if it an event occurrence, each is eligible for removal. REMOVEOCC

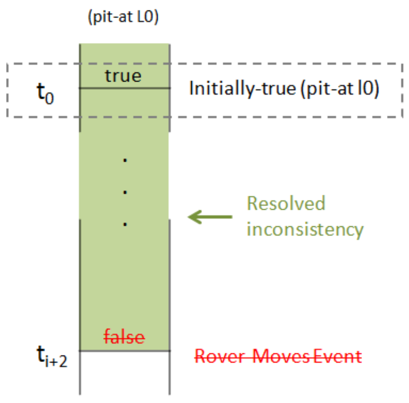


Figure 3: Example of hypothesizing an initial value.

first creates a new set of occurrences, C', by removing o from C. Then, in order to explain why the occurrence does not happen, one of the preconds(e) must be found not to hold at occurrence point t. Therefore, REMOVEOCC creates a new explanation for each precondition p' of e by creating a *removal occurrence* $o_r = (e_r, occ(o))$. The occurrence o_r occurs at the same time, and instead of o; and the new event e_r has no effects and satisfies preconds$(e_r) = \{\neg p'\}$. Due to the presence of the removal event, no new occurrence can be added to the resulting explanation which would cause o to recur.

EXAMPLE 3. *The above mechanism resolves the inconsistency found in example 2 by removing o_{i+2}. Note that no inconsistencies are generated by removing it (recall from example 1 that the rover really didn't move). Finally, the preconditions are examined to look for a precondition which could explain why o_{i+2} might not have occurred. Two preconditions on o_{i+2} are shown in Figure 2(b):* (rover-at r L0) *and* ¬ (pit-at L0). *A new explanation is created for each, with a removal occurrence corresponding to the negated precondition. Each dummy occurrence will cause a new inconsistency, as shown in Figure 2(b).*

3.1.3 Hypothesizing an initial value

Given an inconsistency (p, o, o'), where o refers to the initial observation in the execution history, and p is not observable, a different initial occurrence o may be hypothesized. When this is the case, REFINE generates a new explanation by adding to χ an event occurrence $o_p = (e_p, t_0)$. The event e_p in this occurrence is the reserved initially-true event e_p, which has no preconditions or negative effects, and satisfies effects$^+(e_p) = \{p\}$. This operation has no side effects to any other literal, and thus will never cause new inconsistencies.

EXAMPLE 4. *One of the alternate inconsistencies found in example 3,* (¬ (pit-at L0), occ_0, occ_{i+2})*, has the characteristics required for hypothesizing an initial occurrence. A* pit-at *literal is not observable, and the prior occurrence of the inconsistency is occ_0. Therefore, the discrepancy can be resolved by adding the event occurrence $o_{(pit-atL0)}$, as shown in Figure 3.*

3.2 The FINDEXTRAEVENTS Subroutine

When an explanation no longer has any inconsistencies, it must still be checked for missing events and missing causes. This is due

Algorithm 3: The FINDEXTRAEVENTS Subroutine

```
1 Procedure FINDEXTRAEVENTS ($\chi = (C, R)$)
2 begin
3     foreach $occ_i \in C$ do
4         $occ_j$ ←FINDPROXIMATECAUSE($C$)
5         if $occ_j = \emptyset$ then return $\emptyset$
6     foreach $t_i \in \mathcal{T}$ do
7         $C \leftarrow C \cup$ ENUMERATECAUSEDEVENTS($\chi$)
8     return $\chi$
```

to requirements 2 and 4 of a plausible explanation (see Section 2.3). Requirements 2 and 4 implement the notion of deterministic event execution, by requiring that all events fire immediately when their conditions are met. We now discuss these requirements in more detail.

According to requirement 2, an explanation that includes an event for which no proximate cause is implausible (because the event should have happened earlier). To ensure that such an explanation is not returned, FINDEXTRAEVENTS iterates over each event occurrence occ_i in χ, and attempts to find its proximate cause. This is shown in Algorithm 3, lines 3-5.To do so, it iterates over all occurrences in the explanation and execution history to find the set $PO = [occ_0 \ldots occ_n]$ of occurrences where for every $occ_j \in PO$, $occ_j \prec occ_i$ and there is no occ_k such that $occ_j \prec occ_k \prec occ_i$. If for any $occ_j \in PO$ the set $post(occ_j) \cap pre(occ_i)$ is non-empty, occ_i has a proximate cause. If there is an occ_i that has no proximate cause in the explanation, then the explanation is faulty and a null explanation is returned.

According to requirement 4, all events that are possible must occur. FINDEXTRAEVENTS guarantees this, as shown in Algorithm 3, lines 6-7, by adding events that are not in χ but do have causes in χ. This is done by considering each occurrence point $t_i \in \mathcal{T}$, and enumerating all events e_j whose preconditions are met at time t_i. If an occurrence $occ_ij = (e_j, t_i)$ is not already present in the explanation χ, that occurrence is added to the explanation.

4. DHAGENT

DHAgent is a software agent that operates in planning domains using a PDDL+ [2] domain definition to represent actions, events, and predicates. It conducts planning using the SHOP2 PDDL+ planner [13] and automatically maintains a consistent explanation of the world, which it modifies using the DISCOVERHISTORY algorithm as necessary.

Algorithm 4 shows a high-level description of DHAgent. It takes as input a set of top-level tasks to be performed. It then receives the initial observation, which gives the truth value for the initial state for the observable literals $\mathcal{P}_{obs}$. Because some literals are not observable, many possible worlds may be consistent with this observation. DHAgent creates an initial state according to the closed world assumption, and creates a plan to accomplish the top-level task from this initial state. Because of the closed world assumption, it plans only for the possible world where all unobservable facts are false. DHAgent then loops over the actions in the plan, executing them one at a time, adding projected events to its explanation, and receiving new observations.

When a new observation is inconsistent with its current explanation, DHAgent refines its explanation of the past using DISCOVERHISTORY, as described in Section 3. Although it may sometimes choose an incorrect explanation, it can retract prior assumptions and explained occurrences during subsequent execution steps if they become untenable. Every time the explanation is refined in

Algorithm 4: DHAgent

```
   input: A set $T$ of tasks to perform
1  Procedure DHAgent($T$)
2  begin
3      $\mathsf{obs}_0 \leftarrow$ReceiveObservation()
4      $\pi \leftarrow$Plan($T$, $\mathsf{preconds}(\mathsf{obs}_0)$)
5      $\chi \leftarrow (\{\mathsf{obs}_0\}, \emptyset)$
6      $i \leftarrow 1$
7      while $\pi \neq \emptyset$ do
8          $a_i \leftarrow$Pop($\pi$)
9          $\chi \leftarrow \chi + a_i$
10         $\chi \leftarrow$FindExtraEvents($\chi$)
11         ExecuteAction($a_i$)
12         $\mathsf{obs}_i \leftarrow$ReceiveObservation()
13         $\chi \leftarrow \chi + \mathsf{obs}_i$
14         if $\mathsf{Inconsistencies}(\chi) \neq \emptyset$ then
15             $X_{new} \leftarrow$DiscoverHistory ($\chi$)
16             if $X_{new} \neq \emptyset$ then $\chi \leftarrow$First($X_{new}$)
17             else $\chi \leftarrow (\{\mathsf{obs}_i\}, \emptyset)$
18             Create a set $S$ of all literals satisfied at $\mathsf{occ}(\mathsf{obs}_i)$
               according to $\chi$
19             $p \leftarrow$Plan($T$, $S$)
20         $i \leftarrow i + 1$
```

this way, DHAgent replans, because its current plan may no longer accomplish its task. As part of the explanation refinement process, DHAgent considers the initial world to have been a different possible world consistent with the initial observation. However, it still makes a closed world assumption about the initial state with regard to all literals not hypothesized to have different initial values in the explanation; thus it considers only one possible world at any given time. When no explanations are found, the explanation cannot be maintained further, because there will always be inconsistencies in any explanation based on the current one. Therefore, a new explanation is started as if the most recent observation was the initial observation.

When replanning, DHAgent takes into account the state of the unobservable literals found by projecting the effects of explanation events on the current assumed initial world state. The loop exits only when all actions from a plan have been performed. This occurs either when each top-level task has either been accomplished or has become unachievable.

The way DHAgent makes assumptions about the initial possible world is consistent with the way belief revision is performed in many BDI agents. In the language of belief revision, DHAgent initially believes that all facts in the initial observation are true, and all others are false. Over time it changes its beliefs according to the actions executed and the events in the explanation. Thus DHAgent always believes the state to be the projection of some individual initial world state to the present.

We believe that DHAgent's policy of making strong assumptions about the initial state until observations contradict them is reasonable in domains with many possible worlds and automatic sensing. In such domains, the high number of hidden facts causes planning for all possible worlds to be intractable. Furthermore, DHAgent will be relatively successful in domains where certain literals are rarely true, and cannot be observed directly; the existence of such a true literal would be surprising. For example, although sand pits may be ubiquitous on Mars, a Mars rover could not reason about all possible such pits. Indeed, by assuming sand pits are everywhere it might remain stuck in one place, never moving for fear of falling in. Instead, it initially assumes that the ground is safe and be prepared to revise its assumptions when new evidence arrives. In the same way, DHAgent is an agent that is prepared for surprises, but does not consider every possibility that might occur.

5. EXPERIMENTAL EVALUATION

We examined the performance of DHAgent in the context of planning and execution in two partially-observable domains. These domains have been engineered to include events that are triggered by hidden facts. Thus, events will occur at execution time that can not be predicted due to lack of knowledge at planning time.

Rovers-With-Compass (RWC) is a navigation domain with hidden obstacles inspired by the difficulties encountered by the Mars Rovers. Specifically, individual locations may be windy, sandy, and/or contain sand pits, which the rover cannot observe directly. Sandy locations cause the rover to be covered in sand; while covered in sand, the rover cannot observe its location or perform the "recharge" action. Sand pits stop the rover from moving; the rover can dig itself out at a high energy cost. Windy locations clear the sand off of the rover, but due to a malfunction, may confuse the rover's compass, causing it to move in the wrong direction. When rovers run out of energy, they stop moving.

Satellite-With-Malfunctions (SWM) is based on the Satellites domain from the 2002 International Planning Competition. The objective in each scenario is to acquire images of various phenomena and transmit them to earth. Our additions to this domain include various causes of satellite malfunction: supernova explosions, which can damage sensitive instruments that are pointed toward them; fuel leaks, which cause fuel reserves to diminish rapidly; and motor malfunctions, which delay a satellite's turn to a new perspective. When fuel reserves are depleted, no further goals can be accomplished.

We compared the performance of DHAgent with an ablated version that does not perform explanation, called Non-DHAgent. Non-DHAgent differs in how it replans; replanning occurs whenever the planner incorrectly predicts the new observation. Non-DHAgent then plans based on a state consisting of the latest observation, plus those hidden facts consistent with the events predicted so far by SHOP2. Other systems capable of reasoning about partially observable worlds such as Contingent-FF [5] and SDR[19] provide no support for deterministic exogenous events, or domains where no plan or conditional plan is guaranteed success. To our knowledge, no existing systems can operate in these domains, so we do not compare with existing work.

We wrote a problem generator for each domain that randomly creates an initial state including both observable and hidden facts and goals. For the RWC domain, each starting state contained 3 rovers, and a goal for each rover that required it to move to a new destination. Goal destinations were generated randomly such that the rover must cross at least 3 distinct locations to accomplish its goal. Each scenario took place on a 6×6 grid of locations connected in the four compass directions. Hidden state was assigned independently for each location and condition with probability p.

Each randomly-generated SWM problem included 3 satellites and required the attainment of 8 image acquisition goals. Each image target was chosen randomly from a set of 20. Targets were associated with supernovae, fuel leaks, and motor malfunctions based on a probability p.

We randomly generated 25 problems in each domain; Table 1 shows a comparison of the performance of DHAgent and Non-DHAgent. Here, performance is defined as a percentage of goals completed. For the RWC domain, the probability of hidden difficulties was $p = 0.1$; in the SWM domain, the probability of

Table 1: Statistical t-test results, comparing our explanation-based and non-explanation based systems in the RWC and SWM domains.

Domain	Non-DHAgent	DHAgent	t-test
Rovers	65.3%	78.7%	0.001
Satellite	52.5%	76.0%	< 0.001

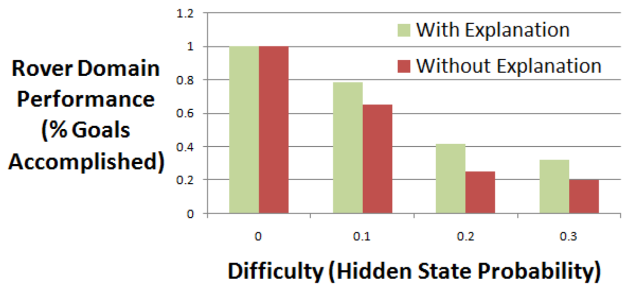


Figure 4: Comparison at various difficulty levels.

hidden state that induces malfunctions was $p = 0.3$. We used a depth bound of 7 in our experiments, i.e., the search for explanations could not include more than 7 recursive refinements.

We compared the mean performances of DHAgent and Non-DHAgent using a two-tailed t-test with paired samples, which showed that DHAgent statistically outperformed Non-DHAgent in both domains. As the only difference between the two agents is the use of explanation, it's clear that the use of DISCOVERHISTORY improved performance. This shows that abductive explanation of state events can improve performance over replanning alone in partially-observable dynamic environments.

To further examine the impact of hidden state on performance, we increased the difficulty of the RWC domain by varying the probability of hidden states. Figure 4 compares the performance of the two agents at 4 difficulty levels: $p = 0.0, 0.1, 0.2$, and 0.3. At $p = 0.0$, there is no hidden state; as expected, we see perfect performance from both agents since no explanation is necessary. As the probability of obstacles rises, both agents perform more poorly, but DHAgent continued to statistically outperform Non-DHAgent. At $p = 0.3$, DHAgent accomplished goals 50% more often than Non-DHAgent. Differences between them were statistically significant for all $p > 0.0$.

6. RELATED WORK

As with many topics in Artificial Intelligence, our work is related to research conducted in several subfields. We've tried to separate the related work by body of literature below, starting with the most closely related first.

Planning and Execution. Other work in planning and execution that involves reconsidering what happened in the past includes that by Molineaux, Klenk, and Aha [12] and Shani and Brafman [19]. Our work extends the work from [12] with a more principled formalism for exogenous events and the capability to reason over a longer history. In more recent work by Shani and Brafman [19], the SDR planner maintains beliefs by reconsidering facts from the past and initial state through a regression process. Unlike our DISCOVERHISTORY algorithm, a series of events that explain the observations is not constructed. Instead, SDR tries to determine what possible facts are consistent by searching paths through a branching and/or tree that covers possible histories tracing through executed sensing actions. SDR considers multiple possible initial states simultaneously to find a plan that works under a sample of possible worlds. The plan will not necessarily execute in the true world, however. When an action's preconditions are not known to be true, the agent replans. In contrast, DHAgent replans when any inconsistency is discovered. As a result of this, SDR will execute some number of actions in an incorrect plan before coming to one that is no longer possible. In contrast, DHAgent never executes actions toward a plan that is inconsistent with its knowledge of the world.

CASPER [10] uses a continuous planning approach to achieve a higher level of responsiveness in dynamic planning situations. The planner has goals, initial state, current state, and a model of the expected future state. The goals or current state may change at any time and invoke the planning process. The planner will then create a new plan based on the current information. This can happen repeatedly and the planner stands ready to continually modify the plan.

Continuous Planning and Execution Framework (CPEF) is introduced in [18]. CPEF assumes that plans are dynamic, that is, that they must be evolving in response to the changes in the environment. Over the years, there has been a large body of research on replanning and plan repair during execution in dynamic environments. These works focus only on execution-time failures as discrepancies; that is, an unexpected state observed during execution triggers a re-planning or plan repair process [24, 9, 1, 22, 21, 18, 17].

In all of the previous works mentioned above, a discrepancy is defined as based on causal links between the preconditions and effects of different actions in the plan. In particular, when the observed state of the world violates such causal-links in the plan, a discrepancy occurs and triggers the re-planning or plan-repair process. In contrast, our formalism and algorithms are designed to generate more expressive and informative explanations of the world, including causal-link failures as well as other changes that may occur due to exogenous events.

Real-Time Control and Planning. Existing research on *real-time control and execution* in Artificial Intelligence typically employs a reactive planning foundation, where the agent decides on an action and executes it immediately [14, 15, 16, **?**, 8]. Sometimes, the systems decide on the action to be executed by using planning heuristics. Sometimes, they generate a complete plan, off-line, that achieves the goal, and then execute the plan.

The Cooperative Intelligent Real-time Control Architecture (CIRCA) is an autonomous planning and control system that builds and executes safety-preserving plans in the face of unpredictable events [14]. CIRCA includes a Reaction Planner which devises a plan to accomplish mission goals while avoiding or preventing failures. The Reaction Planner takes in a problem description that species the initial state of the world, a set of goal states that the planner attempts to reach, a distinguished failure state that the planner must avoid, and a set of transitions that move the world from one state to another. Unlike most planning systems, CIRCA reasons about uncontrollable sources of changes, such as environmental disturbances, failures, and adversaries. The transition models also include timing characteristics that specify how fast the various transitions can occur. The Reaction Planner uses formal verification techniques to check its plans and ensure that failure is not reachable.

Diagnosis. In the diagnosis literature, some work (e.g., [11], [20], [7]) has focused on finding action histories that resolve contradictions by assuming the presence of faulty actions and/or missing assumptions about the initial state. This work differs from ours in that it does not take place in the context of an execution framework, nor does it consider a model of deterministic exogenous events, which requires both simultaneous event occurrence and the elimination of explanations in which caused events fail to occur. Other work in diagnosis (e.g. [6], [4]) has supplemented the diagnosis capability with an execution component; however, when replanning in an execution context, re-explanation may also be necessary, and this earlier work does not support amending earlier explanations by removing events (or actions) previously believed to have happened. Finally, none of these authors has considered the capability of explaining directly based on common planning models as our system does, to reduce the effort required of experts in creating multiple domain models.

7. CONCLUSIONS AND FUTURE WORK

We have described a formalism and algorithm to abductively reason about unexpected event occurrences during planning and execution. The explanations generated using this approach can be used by a planning and execution system to proactively expand its knowledge of the exogenous events and hidden state in the world, and thereby improve the performance of its re-planning process. Our experiments in two planning domains showed that the percentage of goals achieved was significantly higher when using our abductive explanation generation algorithm, compared to an identical system that did not use them. We have shown that this algorithm can improve performance in environments with repeated exposure to hidden events and discrepancies.

There are several tasks that we would like to accomplish in the future based on our results. First, we'll investigate the theoretical properties of abductive reasoning in planning and execution in general. Based on this theory, we intend to generalize our work to investigate explanatory diagnosis in planning with incomplete action and event models, and in temporal planning, where actions and events may have durative effects.

Acknowledgments. This work was supported, in part, by DARPA and the U.S. Army Research Laboratory under contract W911NF-11-C-0037. The views expressed are those of the authors and do not reflect the official policy or position of the funders.

8. REFERENCES

[1] F. Ayan, U. Kuter, F. Yaman, and R. P. Goldman. HOTRiDE: Hierarchical Ordered Task Replanning in Dynamic Environments. In F. Ingrand and K. Rajan, editors, *ICAPS-07 Workshop on Planning and Plan Execution for Real-World Systems*, pages 31–36, 2007.

[2] M. Fox and D. Long. PDDL+: Modelling continuous time-dependent effects. In *Proc. 3rd International NASA Workshop on Planning and Scheduling for Space*, 2002.

[3] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, May 2004.

[4] S. Gspandl, I. Pill, M. Reip, G. Steinbauer, and A. Ferrein. Belief management for high-level robot programs. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[5] J. Hoffmann and R. Brafman. Contingent planning via heuristic forward search with implicit belief states. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS-05)*, pages 71–80, 2005.

[6] G. Iwan. History-based diagnosis templates in the framework of the situation calculus. *KI 2001: Advances in Artificial Intelligence*, pages 244–259, 2001.

[7] G. Iwan and G. Lakemeyer. What observations really tell us. *KI 2003: Advances in Artificial Intelligence*, pages 194–208, 2003.

[8] F. Kabanza, M. Barbeau, and R. St-Denis. Planning control rules for reactive agents. *Artificial Intelligence*, 95(1):67–113, 1997.

[9] S. Kambhampati and J. A. Hendler. A validation-structure-based theory of plan modification and reuse. *Artificial Intelligence*, 55:193–258, 1992.

[10] R. Knight, G. Rabideau, S. Chien, B. Engelhardt, and R. Sherwood. Casper: Space exploration through continuous planning. *IEEE Intelligent System*, pages 70–75, September/October 2001.

[11] S. McIlraith. Explanatory diagnosis: Conjecturing actions to explain observations. In *Proceedings of the Int'l Conference on Principles of Knowledge Representation and Reasoning*, pages 167–179. Morgan Kaufmann Publishers, 1998.

[12] M. Molineaux, M. Klenk, and D. Aha. Goal-driven autonomy in a Navy strategy simulation. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 1548–1554, 2010.

[13] M. Molineaux, M. Klenk, and D. Aha. Planning in dynamic environments: Extending htns with nonlinear continuous effects. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[14] D. Musliner, E. Durfee, and K. Shin. CIRCA: a cooperative intelligent real-time control architecture. *IEEE Transactions on Systems, Man and Cybernetics*, 23(6):1561–1574, 1993.

[15] D. J. Musliner and R. P. Goldman. CIRCA and the Cassini Saturn orbit insertion: Solving a prepositioning problem. In *Working Notes of the NASA Workshop on Planning and Scheduling for Space*, 1997.

[16] D. J. Musliner, M. J. S. Pelican, R. P. Goldman, K. D. Krebsbach, and E. H. Durfee. The evolution of CIRCA, a theory-based AI architecture with real-time performance guarantees, 2008.

[17] K. L. Myers. Advisable planning systems. In A. Tate, editor, *Advanced Planning Technology*. AAAI Press, 1996.

[18] K. L. Myers. A continuous planning and execution framework. *AI Magazine*, 20(4):63, 1999.

[19] G. Shani and R. Brafman. Replanning in domains with partial information and sensing actions. In *Twenty-Second Int'l Joint Conference on Artificial Intelligence*, 2011.

[20] S. Sohrabi, J. Baier, and S. McIlraith. Diagnosis as planning revisited. *Proceedings of the International Conference on the Principles of Knowledge Representation and Reasoning*, pages 26–36, 2010.

[21] X. Wang and S. Chien. Replanning using hierarchical task network and operator-based planning. *Recent Advances in AI Planning*, pages 427–439, 1997.

[22] I. Warfield, C. Hogg, S. Lee-Urban, and H. Munoz-Avila. Adaptation of hierarchical task network plans. In *Proceedings of the Twentieth International FLAIRS Conference (FLAIRS-07)*, 2007.

[23] G. Webster. Nasa's rovers continue martian missions. `http://marsrover.nasa.gov/newsroom/pressreleases/20050524a.html`, May 2005.

[24] S. Yoon, A. Fern, and R. Givan. FF-Replan: A baseline for probabilistic planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pages 352–359, 2007.

Time-Bounded Adaptive A*

Carlos Hernández
Depto. de Ingeniería Informática
Universidad Católica
de la Ssma. Concepción
Caupolican 491, Concepción, Chile
chernan@ucsc.cl

Jorge Baier
Computer Science Department
Pontificia Universidad
Católica de Chile
Santiago, Chile
jabaier@ing.puc.cl

Tansel Uras, Sven Koenig
Computer Science Department
University of
Southern California
Los Angeles, CA 90089, USA
{turas,skoenig}@usc.edu

ABSTRACT

In this paper, we investigate real-time path planning in static terrain, as needed in video games. We introduce the game time model, where time is partitioned into uniform time intervals, an agent can execute one movement during each time interval, and search and movements are done in parallel. The objective is to move the agent from its start location to its goal location in as few time intervals as possible. For known terrain, we show experimentally that Time-Bounded A* (TBA*), an existing real-time search algorithm for undirected terrain, needs fewer time intervals than two state-of-the-art real-time search algorithms and about the same number of time intervals as A*. TBA*, however, cannot be used when the terrain is not known initially. For initially partially or completely unknown terrain, we thus propose a new search algorithm. Our Time-Bounded Adaptive A* (TBAA*) extends TBA* to on-line path planning with the freespace assumption by combining it with Adaptive A*. We prove that TBAA* either moves the agent from its start location to its goal location or detects that this is impossible - an important property since many existing real-time search algorithms are not able to detect efficiently that no path exists. Furthermore, TBAA* can eventually move the agent on a cost-minimal path from its start location to its goal location if it resets the agent into its start location whenever it reaches its goal location. We then show experimentally in initially partially or completely unknown terrain that TBAA* needs fewer time intervals than several state-of-the-art complete and real-time search algorithms and about the same number of time intervals as the best compared complete search algorithm, even though it has the advantage over complete search algorithms that the agent starts to move right away.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Graph and Tree Search Strategies, Heuristic Methods

General Terms

Algorithms, Experimentation

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

Keywords

A*, Incremental Heuristic Search, Learning Real-Time A*, LRTA*, Path Planning with the Freespace Assumption, Real-Time Heuristic Search, TBA*, Time-Bounded A*, Video Games

1. INTRODUCTION

Game characters in video games can execute one movement per game cycle. We therefore introduce the following time model, called game time model. Time is partitioned into uniform time intervals, an agent can execute one movement during each time interval, and search and movements are done in parallel. The objective is to move the agent from its start location to its goal location in as few time intervals as possible. Complete search algorithms, such as A* [4], search first and only then move the agent along the resulting path. They typically need several time intervals to find a complete path from the start location of the agent to its goal location, resulting in a long delay before the agent starts to move and a long time until it reaches its goal location. Real-time search algorithms avoid both issues by executing A* searches and movements in parallel.

Most complete and real-time search algorithms can operate in both known and initially partially or completely unknown terrain. They typically use on-line path planning with the freespace assumption in initially unknown terrain by taking all obstacles into account that the agent has observed so far but assuming that unknown terrain is free of obstacles [11]. For example, Repeated A* is identical to A* except that Repeated A* starts a new A* search from the current location of the agent to its goal location whenever the agent observes obstacles on its current path to its goal location. Incremental search algorithms, such as Adaptive A* [9] and D* Lite [7], behave in the same way but speed up the A* searches by using their experience with prior A* searches to speed up future ones.

In this paper, we study Time-Bounded A* (TBA*) [1]. TBA* is an existing real-time search algorithm for undirected terrain that performs an A* search from the start location of the agent to its goal location. At the end of each time interval, the agent executes a movement towards a location in the OPEN list with the smallest f-value. We show experimentally that TBA* moves the agent in known terrain from its start location to its goal location in fewer time intervals than two state-of-the-art real-time search algorithms and in about the same number of time intervals as A*. However, it cannot be used when the terrain is not known initially. We thus extend it to on-line path planning with the freespace assumption in initially partially or completely unknown (but static) terrain in two steps. In the first step, we extend TBA* to Restarting Time-Bounded A* (RTBA*). RTBA* is identical to TBA* except that, whenever the agent observes obstacles on its current path to a location in the OPEN list with the smallest f-value, RTBA* starts a new A* search from the current location of the agent to its goal location. At the end of each time interval, the agent continues to execute a movement towards a location in the OPEN list with the smallest f-value. In the second step, we extend RTBA* to Time-Bounded Adaptive A* (TBAA*). TBAA* is identical to RTBA* except that TBAA* updates the h-values of the expanded states after each A* search to make them more informed and thus focus future A* searches better, just like Adaptive A* and RTAA* [10]. We prove that RTBA* and TBAA* correctly either move the agent from its start location to its goal location or detect that this is impossible. Many other real-time search algorithms cannot detect efficiently that this is impossible. Furthermore, RTBA* and TBAA* can eventually move the agent on a cost-minimal path from its start location to its goal location if they reset the agent into its start location whenever it reaches its goal location. We show experimentally that TBAA* moves the agent in initially partially or completely unknown terrain from its start location to its goal location in fewer time intervals than several state-of-the-art complete search algorithms (including Adaptive A*) and real-time search algorithms (including RTAA*) and in about the same number of time intervals as the best compared complete search algorithm (namely, D* Lite), even though it has the advantage over complete search algorithms that the agent starts to move right away.

2. GAME TIME MODEL

We now introduce and justify the game time model. Under the game time model, time is partitioned into uniform time intervals, an agent can execute one movement during each time interval, and search and movements are done in parallel. The objective is to move the agent from its start location to its goal location in as few time intervals as possible. The game time model is motivated by video games. Video games often partition time into game cycles, each of which is only a couple of milliseconds long [2]. Each game character executes one movement at the end of each game cycle, which gives the players the illusion of fluid movement. During each game cycle, video games perform all computations necessary to progress the game, which includes the calculation of the next movement of the agent, before redrawing the visuals.

The game time model addresses the fact that the standard way of evaluating search algorithms, namely using their CPU times or path costs, is problematic in real-time situations. A*, for example, needs the smallest CPU time of any search algorithm to find cost-minimal paths (up to tie breaking). Yet, an agent that uses A* might need more time to move from its start location to its goal location than an agent that uses some other search algorithm because A* searches first and only then moves the agent along the resulting path. A* typically needs several time intervals to find a cost-minimal path from the start location of the agent to its goal location, resulting in a long delay before the agent starts to move (which makes the agent unresponsive) and a long time until it reaches its goal location (which makes the agent inefficient) since there is no parallelism of search and movement – the agent does not move until the path is found and does not search afterwards. The advantage of the game time model is that the time needed by the agent to move from its start location to its goal location is proportional to the number of time intervals needed. A search algorithm that computes a path while moving the agent might be able to move the agent to its goal location (along a suboptimal path) in fewer time intervals than A* and thus is more desirable than A*, even though both its CPU time and resulting path cost could be larger than those of A*.

3. NOTATION

A search problem is a tuple $(S, A, c, s_{start}, s_{goal})$, where (S, A) is a finite digraph. S is the set of states, and A is the set of edges. $Succ(s) = \{t \mid (s, t) \in A\}$ is the set of successors (or, synonymously, neighbors) of state $s \in S$. $c : A \mapsto \mathbb{R}^+$ is the cost function that associates a cost with each edge. $s_{start} \in S$ is the start state, and $s_{goal} \in S$ is the goal state. We assume that the digraph is undirected, meaning that there is an edge from vertex s to vertex t iff there is one from t to s, and both edges have the same cost. For simplicity, we call the edges undirected. We also assume that the reader is familiar with A* and knows that A*, when searching from s_{root} to s_{goal}, maintains an OPEN list (a priority queue) and, for every state $s \in S$, a g-value $g(s)$ that is the cost of the cost-minimal path from s_{root} to s found so far, an h-value $h(s)$ that is an approximation of the cost of a cost-minimal path from s to s_{goal}, an f-value $f(s) = g(s) + h(s)$, and a parent pointer *parent*(s) that points to the parent of s in the A* search tree, whose root is s_{root}. $H(s)$ is the user-provided h-value of s. The h-values are consistent iff $h(s_{goal}) = 0$ and $h(s) \leq c(s, t) + h(t)$ for all states $s \in S$ and $t \in Succ(s)$.

4. TBA*

Given a search problem, the objective is to move an agent from s_{start} to s_{goal} in as few time intervals as possible. Real-time search algorithms execute one or more A* searches to determine one movement for the agent per time interval, that is, after a bounded amount of computation. Time-Bounded A* (TBA*) [1] is an existing real-time search algorithm for search problems with known cost functions. We use known four-neighbor grids with blocked and unblocked cells as examples, where the states are the cells and edges connect every unblocked cell to its unblocked neighboring cells to the north, east, south and west with cost one. TBA* performs an A* search from $s_{root} = s_{start}$ to s_{goal}. At the end of each time interval, the agent executes the movement from its current state $s_{current}$ towards a state s_{best} in the OPEN list with the smallest f-value (where s_{goal} should be chosen if possible), as follows: Consider the branch *path* of the A* search tree from its root $s_{root} = s_{start}$ to s_{best}, which is a cost-minimal path from s_{root} to s_{best}. If $s_{current}$ is on *path* (that is, $s_{current}$ is encountered when following the parent pointers from s_{best} to s_{root}), then the agent executes the movement from $s_{current}$ towards s_{best} along *path* (that is, moves to the state after $s_{current}$ on *path*). Otherwise, it executes the movement from $s_{current}$ towards s_{root} along the branch of the A* search tree from s_{root} to $s_{current}$ (that is, follows the parent pointer of $s_{current}$), which is possible since the edges are undirected. The time that it takes to expand a state depends on the size of the OPEN list, and

Algorithm 1 TBA*

```
 1: procedure InitializeState($s$)
 2: if $search(s) = 0$ then
 3:     $h(s) := H(s)$;
 4:     $g(s) := \infty$;
 5: $search(s) := searchnumber$;

 6: procedure InitializeSearch()
 7: $searchnumber := searchnumber + 1$;
 8: $s_{root} := s_{current}$;
 9: InitializeState($s_{root}$);
10: $g(s_{root}) := 0$;
11: $OPEN := \emptyset$;
12: Insert $s_{root}$ into $OPEN$;
13: InitializeState($s_{goal}$);
14: $goalFoundFlag := 0$;

15: function Search()
16: $expansions := 0$;
17: while $OPEN \neq \emptyset$ AND $expansions < k$ AND
18:         $g(s_{goal}) + h(s_{goal}) > \min_{t \in OPEN}(g(t) + h(t))$ do
19:     $s := \arg\min_{t \in OPEN}(g(t) + h(t))$;
20:     Remove $s$ from $OPEN$;
21:     for all $t \in Succ(s)$ do
22:         InitializeState($t$)
23:         if $g(t) > g(s) + c(s,t)$ then
24:             $g(t) := g(s) + c(s,t)$;
25:             $parent(t) := s$;
26:             Insert $t$ into $OPEN$;
27:     $expansions := expansions + 1$;
28: if $OPEN = \emptyset$ then
29:     return false;
30: $s_{best} := \arg\min_{t \in OPEN}(g(t) + h(t))$;
31: if $s_{best} = s_{goal}$ then
32:     $goalFoundFlag := 1$;
33: $path :=$ path from $s_{root}$ to $s_{best}$;
34: return true;

35: function MoveToGoal()
36: $s_{current} := s_{start}$;
37: InitializeSearch();
38: while $s_{current} \neq s_{goal}$ do
39:     print("a new time interval starts now");
40:     if $goalFoundFlag = 0$ then
41:         if Search() = false then
42:             return false;
43:     if $s_{current}$ is on $path$ then
44:         $s_{current} :=$ state after $s_{current}$ on $path$;
45:     else
46:         $s_{current} := parent(s_{current})$;
47:     Execute movement to $s_{current}$;
48: return true;

49: procedure Main()
50: $searchnumber := 0$;
51: for all $s \in S$ do
52:     $search(s) := 0$;
53: if MoveToGoal() = true then
54:     print("the agent is now at the goal state");
55: else
56:     print("the agent cannot reach the goal state");
```

the time that it takes to determine the next movement for the agent depends on the length of *path*. In the following, we make the simplifying assumption that the time per state expansion is constant and the time per movement determination is zero, resulting in a constant number k of state expansions during each time interval. The original version of TBA* uses an additional parameter and techniques that make it a true (amortized) real-time search algorithm. We could do the same for RTBA* and TBAA* but will not do so for simplicity.

Algorithm 1 shows TBA*. Procedure InitializeState initializes the h-value of a given state (to the user-provided h-value) and the g-value of the state (to infinity) when they are needed for the first time, to avoid initializing those values that are not needed later. *searchnumber* is the number of the current A* search, and *search*(s) is the number of the A* search during which the g-value of state s was initialized last. It is zero if the g-value has not been initialized yet. Procedure Search performs an A* search from s_{root} to s_{goal} by expanding states until the OPEN list becomes empty (in which case the agent cannot reach s_{goal}), until a cost-minimal path from s_{root} to s_{goal} has been found (that is, $s_{best} = s_{goal}$) or until k states have been expanded. *expansions* is the number of expanded states in the current time interval, and *goalFoundFlag* is true iff the A* search has found a cost-minimal path from s_{root} to s_{goal}. Procedure Search sets *goalFoundFlag* and *path* before it terminates. Procedure MoveToGoal sets the agent into s_{start} and calls procedure InitializeSearch to initialize the A* search from $s_{root} = s_{current}$ to s_{goal}. It then repeatedly calls procedure Search (if *goalFoundFlag* is false) and then executes a movement on Lines 43-47, until the agent reaches s_{goal} or the A* search indicates that the agent cannot reach s_{goal}. Procedure Main calls procedure MoveToGoal for a given search problem and reports the results.

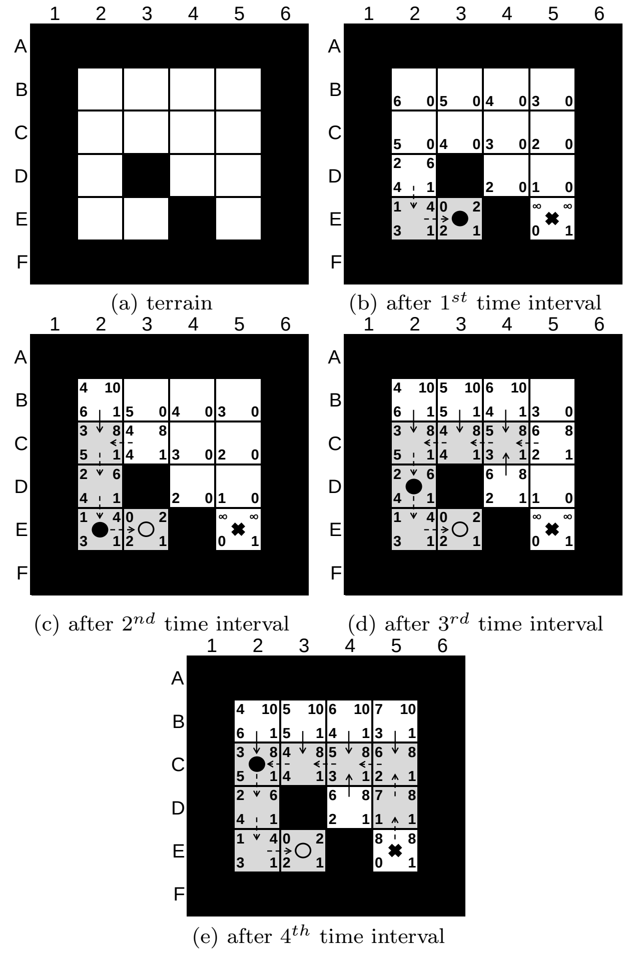

(a) terrain
(b) after 1st time interval
(c) after 2nd time interval
(d) after 3rd time interval
(e) after 4th time interval

Figure 1: Operation of TBA*

Figure 1 shows the operation of TBA* with $k = 2$ in the known four-neighbor grid with blocked (black) and unblocked (white) cells shown in Figure 1(a). $s_{start} = E3$, and $s_{goal} = E5$. The user-provided h-values are the Manhattan distances, that is, the costs of cost-minimal paths from the cells to the goal cell on a four-neighbor grid without blocked cells. The OPEN list contains the non-expanded cells that have a neighboring expanded (grey) cell. The bottom-left and bottom-right corners of each cell show its h- and search-values, respectively. The top-left and top-right corners of each cell show its g- and f-values, respectively, iff its g-value has been initialized. The arrow shows its parent pointer iff its parent pointer has been initialized. Dotted arrows indicate the cost-minimal path from $s_{root} = s_{start}$ to s_{best}. Figure 1(b) shows the situation at the end of the first time interval (but before the first movement of the agent). The OPEN list contains only $D2$ with f-value 6. $s_{current} = E3$ is on the branch *path* of the A* search tree from $s_{root} = E3$ to $s_{best} = D2$. The agent thus executes the movement along this branch to $E2$. Figure 1(c) shows the situation at the end of the second time interval (but before the second movement of the agent). The OPEN list contains $C3$ with f-value 8 and $B2$ with f-value 10. $s_{current} = E2$ is on the branch *path* of the A* search tree from $s_{root} = E3$ to $s_{best} = C3$. The agent thus executes the movement along this branch to $D2$, and so on. Figure 1(e) shows that the A* search finds a cost-minimal path from $s_{root} = E3$ to $s_{goal} = E5$ at the end of the fourth time interval (but before the fourth movement of the agent). The agent then executes movements along this branch without any further cell expansions until it reaches $s_{goal} = E5$.

5. RTBA*

TBA* does not apply to search problems with unknown cost functions. We require that the agent observes the cost of an edge before it traverses it. In cases where a lower bound on the cost function is known, search algorithms can use the lower bound when an edge cost is unknown. This way, the observation of an actual edge cost can change the assumed edge cost only once, namely from the lower bound to the actual edge cost, which cannot decrease the assumed edge cost - properties that we exploit in the following.[1] We use initially unknown four-neighbor grids with blocked and unblocked cells as examples. The agent knows its start and goal cells. It does not know the blockage status of the cells initially but always observes the blockage status of its four neighboring cells to the north, east, south and west. It can use path planning with the freespace assumption by assuming that all cells are unblocked, that is, edges connect every cell to its neighboring cells to the north, east, south and west with cost one. If, after executing a movement, it observes that a neighboring cell is blocked, it increases the costs of all incoming and outgoing edges of that cell to infinity, which is equivalent to removing the edges from the set of edges A. The agent could run TBA* again whenever edge costs have increased during the current A* search but would then often abandon the current A* search unnecessarily. Instead, it runs TBA* again only when edge costs on the path from $s_{current}$ to s_{best} have increased during the current A* search. We refer to this search algorithm as Restarting Time-Bounded A* (RTBA*).

Algorithm 2 shows RTBA* without repeating the InitializeSearch, Search and Main procedures from Algorithm 1. Procedure InitializeState now initializes the h-value of a

[1]We just refer to edge costs and the context determines whether we mean the actual or assumed edge costs.

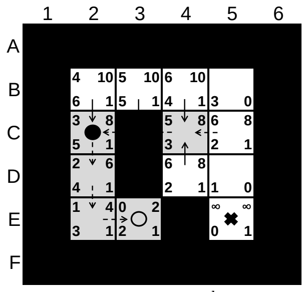
Algorithm 2 RTBA*

```
1: procedure InitializeState($s$)
2: if $search(s) = 0$ then
3:     $h(s) := H(s)$;
4:     $g(s) := \infty$;
5: else if $search(s) \neq searchnumber$ then
6:     $g(s) := \infty$;
7: $search(s) := searchnumber$;

8: procedure StartNewSearch?()
9: if edge costs on $path$ have increased
10:             since the last call to InitializeSearch then
11:     InitializeSearch();
12:     $path := empty$;

13: function MoveToGoal()
14: $s_{current} := s_{start}$;
15: InitializeSearch();
16: Observe edge cost increases (if any);
17: while $s_{current} \neq s_{goal}$ do
18:     print("a new time interval starts now");
19:     if $goalFoundFlag = 0$ then
20:         if Search() = false then
21:             return false;
22:     StartNewSearch?();
23:     if $path \neq empty$ then
24:         if $s_{current}$ is on $path$ then
25:             $s_{current}$ := state after $s_{current}$ on $path$;
26:         else
27:             $s_{current} := parent(s_{current})$;
28:         Execute movement to $s_{current}$;
29:         Observe edge cost increases (if any);
30:         StartNewSearch?();
31: return true;
```

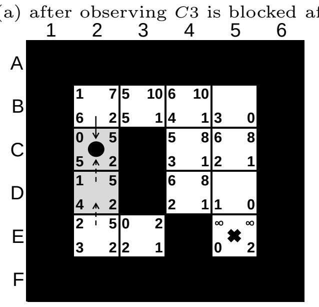
given state (to the user-provided h-value) when it is needed for the first time. It initializes the g-value of the state (to infinity) when it is needed by the current A* search for the first time. Procedure StartNewSearch? checks whether edge costs on the path from $s_{current}$ to s_{best} have increased during the current A* search and, if so, starts a new A* search from $s_{root} = s_{current}$ to s_{goal}.[2] Procedure MoveToGoal now calls procedure StartNewSearch? on Line 30 after the agent executes a movement since the agent might observe additional edge costs which result in increases of edge costs on the path from $s_{current}$ to s_{best}. Procedure MoveToGoal also calls procedure StartNewSearch? on Line 22 after the call to procedure Search since it might result in a new path from $s_{current}$ to s_{best} that contains edge costs increases that have been ignored earlier.

Figure 2(a,b,d) shows the operation of RTBA* with $k = 2$ in an initially unknown four-neighbor grid with blocked and unblocked cells. $s_{start} = E3$, and $s_{goal} = E5$. The user-provided h-values are the Manhattan distances. The first three time intervals are as in Figure 1 except that the agent observes blocked cell $C3$ after the third time interval and movement of the agent, which increases edge costs on the

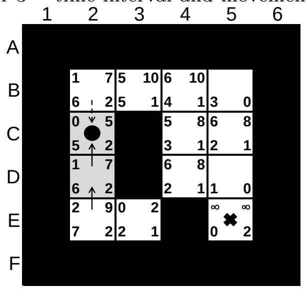
[2]Actually, procedure StartNewSearch? checks the branch *path* of the A* search tree from s_{root} to s_{best} rather than the path from $s_{current}$ to s_{best}. The agent traverses only edges that belong to branches of the A* search tree. Consider the branches of the A* search tree from s_{root} to $s_{current}$ and from s_{root} to s_{best}. There cannot be any edge cost increases on the branch from s_{root} to $s_{current}$ since the edges are undirected and the agent has traversed the edges of the branch earlier already. The path from $s_{current}$ to s_{best} is made up of both branches without their common prefix. Thus, edge costs increase on the path iff they increase on the branch from s_{root} to s_{best}.



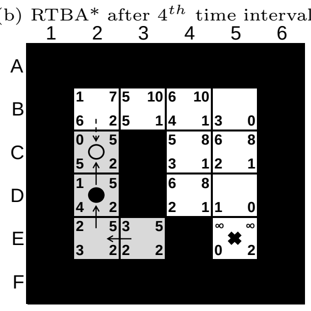
(a) after observing $C3$ is blocked after 3rd time interval and movement



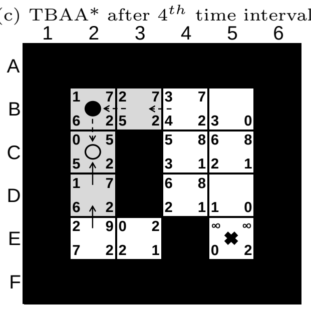
(b) RTBA* after 4th time interval



(c) TBAA* after 4th time interval



(d) RTBA* after 5th time interval



(e) TBAA* after 5th time interval

Figure 2: Operation of RTBA* and TBAA*

path from $s_{current} = C2$ to $s_{goal} = E5$. Thus, RTBA* starts a new A* search from $s_{root} = s_{current} = C2$ to $s_{goal} = E6$. Figure 2(b) shows the situation at the end of the fourth time interval (but before the fourth movement of the agent). The OPEN list contains $E2$ with f-value 5 and $B2$ with f-value 7. $s_{current} = C2$ is on the branch *path* of the A* search tree from $s_{root} = C2$ to $s_{best} = E2$. The agent thus executes the movement along this branch to $D2$. Figure 2(d) shows the situation at the end of the fifth time interval (but before the fifth movement of the agent). The OPEN list contains only $B2$ with f-value 7. $s_{current} = D2$ is not on the branch *path* of the A* search tree from $s_{root} = C2$ to $s_{best} = B2$. The agent thus executes the movement that follows the parent pointer of $s_{current} = D2$ to $C2$.

6. TBAA*

Each time RTBA* starts a new A* search, all information from the previous A* search is lost. However, real-time search algorithms often update the h-values to make them more informed. We therefore propose Time-Bounded Adaptive A* (TBAA*). TBAA* works like RTBA* but updates h-values of cells in the way (Lazy) Adaptive A* [9] and (Lazy) RTAA* do. Each time TBAA* starts a new A* search, it updates the h-values of all generated states s by the previous A* search by assigning $h(s) := f(s_{best}) - g(s)$ if this increases $h(s)$, which maintains the consistency of the

Algorithm 3 TBAA*

```
1: procedure InitializeState(s)
2: if $search(s) = 0$ then
3:     $h(s) := H(s)$;
4:     $g(s) := \infty$;
5: else if $search(s) \neq searchnumber$ then
6:     if $h(s) < pathcost(search(s)) - g(s)$ then
7:         $h(s) := pathcost(search(s)) - g(s)$;
8:     $g(s) := \infty$;
9: $search(s) := searchnumber$;

10: procedure StartNewSearch?()
11: if edge costs on path have increased
12:         since the last call to InitializeSearch then
13:     $pathcost(searchnumber) := \min_{s \in OPEN}(g(s) + h(s))$;
14:     InitializeSearch();
15:     $path := empty$;
```

h-values. TBAA* performs this h-value update only once the h-value of a state is needed by the current A* search for the first time, to avoid computing those h-values that are not needed later.

Algorithm 3 shows TBAA* without repeating the InitializeSearch, Search and Main procedures from Algorithm 1 and the MoveToGoal procedure from Algorithm 2. Procedure StartNewSearch? now remembers $f(s_{best})$ on Line 13 by assigning $pathcost(searchnumber) := f(s_{best})$, and Procedure InitializeState now updates the h-value of the given state s on Line 7 when it is needed by the current A* search for the first time by assigning $h(s) := pathcost(search(s)) - g(s)$ if this increases $h(s)$.

Figure 2(a,c,e) show the operation of TBAA* with $k = 2$ in the same scenario as described for RTBA*. TBAA* sets $pathcost(1) = f(C5) = 8$ and starts a new A* search from $s_{root} = s_{current} = C2$ to $s_{goal} = E6$. Figure 2(c) shows the situation at the end of the fourth time interval (but before the fourth movement of the agent). The OPEN list contains $E2$ with f-value 9 (four higher than for RTBA*) and $B2$ with f-value 7. $s_{current} = C2$ is on the branch *path* of the A* search tree from $s_{root} = C2$ to $s_{best} = B2$. The agent thus executes the movement along this branch to $B2$ and needs fewer movements to reach $s_{goal} = E5$ than RTBA*, demonstrating the advantage of updating the h-values.

7. ANALYSIS

We now prove that RTBA* and TBAA* correctly either move the agent from s_{start} to s_{goal} or detect that this is impossible. Furthermore, RTBA* and TBAA* can eventually move the agent on a cost-minimal path from s_{start} to s_{goal} if they reset the agent into s_{start} whenever it reaches s_{goal}. We make use of the following assumptions and properties: The user-provided h-values are consistent. The h-value updates of Adaptive A* (and thus also the ones of TBAA*) maintain the consistency of the h-values [9]. An A* search with consistent h-values correctly finds a path or detects that none exists [12], and the found path is cost-minimal [12].

THEOREM 1. *RTBA* and TBAA* correctly either move the agent from s_{start} to s_{goal} or detect that this is impossible.*

PROOF. Each time the search algorithm starts a new A* search, it has observed at least one additional edge cost. Thus, the number of times it starts a new A* search is bounded. Consider the last A* search, and let s_{root} be the root of the A* search tree. This A* search correctly finds a

Algorithm 4 Repeated Runs of RTBA* and TBAA*

```
1: procedure Main()
2: $searchnumber := 0$;
3: for all $s \in S$ do
4:     $search(s) := 0$;
5: repeat
6:     if MoveToGoal() = false then
7:         print("the agent cannot reach the goal state");
8: until $s_{root} = s_{start}$;
9: print("cost-minimal path from start state to goal state: ");
10: print($path$);
```

path from s_{root} to s_{goal} or detects that none exists. There exists a path from s_{root} to s_{goal} iff there exists a path from s_{start} to s_{goal} since the agent moves along undirected edges. If the A* search finds a path from s_{root} to s_{goal}, then the agent repeatedly follows the parent pointers of its current states until it is on the path and then traverses the path to s_{goal}. The agent is able to follow the parent pointers since edges are undirected and the agent has traversed the edges earlier already during the A* search. The agent reaches the path this way since the agent moves in the A* search tree and thus reaches s_{root} by repeatedly following the parent pointers if it does not reach the path from s_{root} to s_{goal} earlier already. The agent is able to traverse the path to s_{goal} since the A* search correctly found a path from s_{root} to s_{goal} and the search algorithm would have started a new A* search if edge costs on the path had increased during the A* search. □

The following theorem states that RTBA* and TBAA* eventually find a cost-minimal path from s_{start} to s_{goal} if they reset the agent into s_{start} whenever it reaches s_{goal}. Algorithm 4 defines this process precisely.

THEOREM 2. *Algorithm 4 prints a cost-minimal path for RTBA* and TBAA* if a path from s_{start} to s_{goal} exists.*

PROOF. Each time after the search algorithm resets the agent into s_{start} and starts a new A* search, it moves the agent from s_{start} to s_{goal} according to Theorem 1. Each time the search algorithm starts a new A* search while the agent moves to s_{goal}, it has observed at least one additional edge cost. Thus, the number of times it starts a new A* search while the agent moves to s_{goal} is bounded, and it eventually moves the agent from s_{start} to s_{goal} without starting a new A* search while the agent moves to s_{goal} (implying that $s_{root} = s_{start}$). The last A* search finds a cost-minimal path from s_{start} to s_{goal}, and there cannot be any edge cost increases on the path since edges are undirected and the agent has traversed the edges earlier already during the A* search. The agent thus traverses a cost-minimal path if it moves along this path. □

8. EXPERIMENTAL EVALUATION

We compare RTBA* and TBAA* against several state-of-the-art complete and real-time search algorithms, which we implemented in a similar way, for example, using a standard binary heap for the OPEN list and breaking ties among states with the same f-value in favor of larger g-values. All complete search algorithms first find a complete path for the agent from the start state to the goal state (over several time intervals) and then move the agent along it (again over several time intervals). All real-time search algorithms perform only state expansions during the first time interval. In each subsequent time interval, they compute a path for the agent, execute one movement for the agent, and then perform repeatedly state expansions until the end of the time interval is reached, implying that all time intervals have approximately the same length but not necessarily the same number of state expansions. All real-time search algorithms perform an h-value update only once the h-value of a state is needed by the current A* search for the first time. The scaling behavior of the search algorithms is less important than the hardware and implementation details since the search problems are small. It is difficult to compare the search algorithms with proxies, such as the number of state expansions, instead of the runtime itself since they perform different basic operations. We thus do not know of any better method for evaluating them than to implement them as best as possible and let other researchers validate the results with their own and thus potentially slightly different implementations.

We use known and initially partially or completely unknown eight-neighbor grids with blocked and unblocked cells in the experiments. Four-neighbor grids make for good illustrations since they result in integer-valued g-, h-, and f-values but we believe that eight-neighbor grids are more realistic for video games [3]. Also, some incremental search algorithms speed up A* searches less on eight-neighbor grids. The user-provided h-values are the octile distances, that is, the costs of cost-minimal paths from the cells to the goal cell on an eight-neighbor grid without blocked cells. The agent knows the dimensions of the grid and its start and goal cells. It can always move from its current unblocked cell to one of the eight unblocked neighboring cells with cost one for horizontal or vertical movements and cost $\sqrt{2}$ for diagonal movements. We ran the search algorithms with time intervals whose lengths ranged from 0.3 to 1.5 milliseconds and report results for the average number of time intervals and number of movements until the agent reaches the goal cell for the first time.

We use six game maps and generated 300 search problems with randomly chosen start and goal cells for each game map, for a total of 1,800 search problems.[3] In known grids, it knows the blockage status of all cells initially. In initially partially or completely unknown grids, it does not know the blockage status of some or all, respectively, cells initially but always observes the blockage status of its eight neighboring cells. Partially unknown grids are motivated by video games where the layout of the terrain is known to the agent but other players can build structures which are initially unknown to the agent [13]. We randomly blocked 15 percent of the unblocked cells in the game maps. The agent knows the blockage status of all blocked cells in the game maps but does not know the blockage status of the additional blocked cells.

8.1 Known Terrain

In known terrain, we compare TBA* against the complete search algorithm (forward) A* and the real-time search algorithms RTAA* and daRTAA*. Table 1 shows the following

[3]We use three Starcraft maps, namely Enigma of size 768×768, Inferno of size 768×768, and WheelofWar of size 768×768. We use three Dragon Age: Origins maps, namely orz103d of size 456×463, orz702d of size 939×718, and orz703d of size 502×652.

Table 1: Known Terrain

	RTAA*		daRTAA*		TBA*		A*	
Length of Time Intervals (ms)	# Time Intervals	# Movements	# Time Intervals	# Movements	# Time Intervals	# Movements	# Time Intervals	# Movements
0.3	2,193	2,192	1,729	1,728	**568**	567	584	545
0.6	1,541	1,540	1,303	1,302	**556**	555	565	545
0.9	1,362	1,361	1,151	1,150	**552**	551	558	545
1.2	1,196	1,195	1,057	1,056	**550**	549	555	545
1.5	1,087	1,086	970	969	**549**	548	553	545

Table 2: Initially Completely Unknown Terrain

	RTAA*		daRTAA*		RTBA*		TBAA*		Repeated A*		Adaptive A*		D* Lite	
Length of Time Intervals (ms)	# Time Intervals	# Movements	# Time Intervals	# Movements	# Time Intervals	# Movements	# Time Intervals	# Movements	# Time Intervals	# Movements	# Time Intervals	# Movements	# Time Intervals	# Movements
0.3	3,245	3,244	2,879	2,878	4,613	4,604	2,290	2,286	7,155	2,004	3,230	2,010	**2,203**	2,027
0.6	2,598	2,597	2,472	2,471	3,368	3,360	2,147	2,144	4,487	2,004	2,572	2,010	**2,090**	2,027
0.9	2,451	2,450	2,418	2,417	2,918	2,910	2,101	2,099	3,611	2,004	2,361	2,010	**2,062**	2,027
1.2	2,310	2,309	2,305	2,304	2,695	2,688	2,086	2,083	3,178	2,004	2,260	2,010	**2,051**	2,027
1.5	2,281	2,280	2,272	2,271	2,560	2,553	2,070	2,068	2,920	2,004	2,202	2,010	**2,045**	2,027

Table 3: Initially Partially Unknown Terrain

	RTAA*		daRTAA*		RTBA*		TBAA*		Repeated A*		Adaptive A*		D* Lite	
Length of Time Intervals (ms)	# Time Intervals	# Movements	# Time Intervals	# Movements	# Time Intervals	# Movements	# Time Intervals	# Movements	# Time Intervals	# Movements	# Time Intervals	# Movements	# Time Intervals	# Movements
0.3	2,694	2,693	2,460	2,459	2,734	2,730	**1,505**	1,504	6,324	1,409	2,430	1,399	1,659	1,418
0.6	2,039	2,038	1,863	1,862	2,037	2,034	**1,442**	1,441	3,812	1,409	1,875	1,399	1,532	1,418
0.9	1,840	1,839	1,779	1,778	1,860	1,857	**1,431**	1,430	2,979	1,409	1,695	1,399	1,490	1,418
1.2	1,707	1,706	1,643	1,642	1,726	1,724	**1,421**	1,420	2,564	1,409	1,608	1,399	1,470	1,418
1.5	1,620	1,619	1,642	1,641	1,668	1,666	**1,415**	1,414	2,316	1,409	1,556	1,399	1,458	1,418

relationships for known terrain.

- The average number of movements until the agent reaches the goal cell does not depend on the length of the time intervals for A* since complete search algorithms search first and only then move the agent along the resulting path.
- The average number of movements until the agent reaches the goal cell decrease as the length of the time intervals increases for TBA*, RTAA*, and daRTAA*.
- The average number of time intervals until the agent reaches the goal cell decreases as the length of the time intervals increases for all search algorithms.

All real-time search algorithms move the agent in known terrain from its start cell to its goal cell in about the same or more time intervals than A*. However, TBA* moves the agent in known terrain from its start cell to its goal cell in fewer time intervals than the two real-time search algorithms RTAA* and daRTAA* and in about the same number of time intervals as A*. TBA* has the advantage over A* that the agent moves right away for TBA* (namely, during time interval 2 in our implementation) rather than only in time intervals 8 to 40 on average for A*.

8.2 Initially Unknown Terrain

In initially unknown terrain, we compare RTBA* and TBAA* against the complete search algorithms (forward) Repeated A*, Adaptive A*, and D* Lite and the real-time search algorithms RTAA* and daRTAA*. Our implementation of Repeated A* starts a new A* search only when edge costs on the path from $s_{current}$ to s_{goal} have increased, rather than whenever edge costs have increased, different from [8], which explains the difference in experimental results compared to [8]. Tables 2 and 3 show the following relationships for initially completely and partially unknown terrain, respectively.

- The average number of movements until the agent reaches the goal cell does not depend on the length of the time intervals for Repeated A*, Adaptive A* and D* Lite. Furthermore, they are similar for all search algorithms, since they execute the same movements (modulo tie breaking).
- The average number of movements until the agent reaches the goal cell decreases as the length of the time intervals increases for RTBA*, TBAA*, RTAA*, and daRTAA*.
- The average number of time intervals until the agent reaches the goal cell decreases as the length of the time intervals increases for all search algorithms.

All real-time search algorithms move the agent in initially partially or completely unknown terrain from its start cell to its goal cell in fewer time intervals than Repeated A*. The game time model is thus able to explain the importance of real-time search in this case. TBAA* moves the agent in initially partially or completely unknown terrain from its start cell to its goal cell in fewer time intervals than the two complete search algorithms Repeated A* and Adaptive A*, and the two real-time search algorithms RTAA* and daRTAA* and in about the same number of time intervals as the best compared complete search algorithm D* Lite. TBAA* seems to have a slight advantage over D* Lite in initially partially unknown terrain and vice versa in initially completely unknown terrain (although this difference might not be statistically significant). The reason appears to be that the h-value surface does not have local minima on grids without blocked cells, which allows D* Lite to speed up A* searches significantly during the first searches in initially completely unknown terrain but not in initially partially unknown terrain. For example, D* Lite is often able to find the very first path in initially completely unknown terrain in one time interval.

9. SUMMARY

In this paper, we introduced the game time model, where time is partitioned into uniform time intervals, an agent can execute one action during each time interval, and search and action execution run in parallel. We then extended Time-Bounded A* (TBA*) to on-line path planning with the freespace assumption in initially partially or completely unknown (but static) terrain, resulting in Time-Bounded Adaptive A* (TBAA*). Similar to TBA*, TBAA* performs an A* search from the start location of the agent to its goal location. At the end of each time interval, the agent executes a movement towards a location in the OPEN list with the smallest f-value. TBAA* starts a new A* search whenever the agent observes obstacles on its path to this location. Similar to Adaptive A*, TBAA* updates the h-values of the expanded states after each A* search to make them more informed and thus focus future A* searches better. We proved that TBAA* correctly either moves the agent from its start location to its goal location or detects that this is impossible. Many other real-time search algorithms cannot detect efficiently that no path exists. Furthermore, TBAA* can eventually move the agent on a cost-minimal path from its start location to its goal location if it resets the agent into its start location whenever it reaches its goal location. We then showed experimentally that TBAA* moves the agent in initially partially or completely unknown terrain from its start location to its goal location in fewer time intervals than several complete and real-time search algorithms and in about the same number of time intervals as the best compared complete search algorithm, even though it has the advantage over complete search algorithms that the agent starts to move right away. In future work, we intend to let TBAA* use techniques that speed up daRTAA* over RTAA* [6, 5] to avoid local minima in the h-value surface. Since TBAA* uses less than 30 percent of the available time on average, we also intend to let it use more sophisticated h-value update techniques to make its h-values even more informed.

Acknowledgments

We thank Nathan Sturtevant for making his game maps publicly available at www.movingai.com. We also thank the anonymous reviewers, whose comments resulted in substantial revisions of our original manuscript. This material is based upon research supported by NSF (while Sven Koenig was serving at NSF). It is also based upon research supported by ARL/ARO under contract/grant number W911NF-08-1-0468 and ONR in form of a MURI under contract/grant number N00014-09-1-1031. The research was performed while Carlos Hernández visited the University of Southern California. Jorge Baier was partly funded by Fondecyt grant number 11110321. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies or the U.S. government.

10. REFERENCES

[1] Yngvi Björnsson, Vadim Bulitko, and Nathan Sturtevant. TBA*: Time-bounded A*. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 431–436, 2009.

[2] Vadim Bulitko, Yngvi Björnsson, Nathan Sturtevant, and Ramon Lawrence. *Real-time Heuristic Search for Pathfinding in Video Games.* available from: https://sites.google.com/a/ualberta.ca/ircl/projects/rths.

[3] Vadim Bulitko and Greg Lee. Learning in real time search: a unifying framework. *Journal of Artificial Intelligence Research*, 25:119–157, 2006.

[4] Peter Hart, Nils Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimal cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[5] Carlos Hernández and Jorge Baier. Real-Time Adaptive A* with depression avoidance. In *Proceedings of the 7th International Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 146–151, 2011.

[6] Carlos Hernández and Jorge Baier. Real-time heuristic search with depression avoidance. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 578–583, 2011.

[7] Sven Koenig and Maxim Likhachev. D* Lite. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI)*, pages 476–483, 2002.

[8] Sven Koenig and Maxim Likhachev. Fast replanning for navigation in unknown terrain. *Transactions on Robotics*, 21(3):354–363, 2005.

[9] Sven Koenig and Maxim Likhachev. A new principle for incremental heuristic search: Theoretical results. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 402–405, 2006.

[10] Sven Koenig and Maxim Likhachev. Real-Time Adaptive A*. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 281–288, 2006.

[11] Sven Koenig, Craig Tovey, and Yury Smirnov. Performance bounds for planning in unknown terrain. *Artificial Intelligence*, 147(1-2):253–279, 2003.

[12] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, Boston (Massachusetts), 1984.

[13] Peter Yap, Neil Burch, Robert Holte, and Jonathan Schaeffer. Any-angle path planning for computer games. In *Proceedings of the 7th International Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 2011.

Session 2F
Knowledge Representation & Reasoning

Memory Formation, Consolidation, and Forgetting in Learning Agents

Budhitama Subagdja,
Wenwen Wang, Ah-Hwee Tan
School of Computer Engineering,
Nanyang Technological University
Nanyang Avenue, Singapore 639798
{budhitama,wa0003en,asahtan}@ntu.edu.sg

Yuan-Sin Tan, Loo-Nin Teow
DSO National Laboratories,
20 Science Park Drive, Singapore 118230
{tyuansin,tloonin}@dso.org.sg

ABSTRACT

Memory enables past experiences to be remembered and acquired as useful knowledge to support decision making, especially when perception and computational resources are limited. This paper presents a neuropsychological-inspired dual memory model for agents, consisting of an episodic memory that records the agent's experience in real time and a semantic memory that captures factual knowledge through a parallel consolidation process. In addition, the model incorporates a natural forgetting mechanism that prevents memory overloading by removing transient memory traces. Our experimental study based on a real-time first-person-shooter video game has indicated that the memory consolidation and forgetting processes are not only able to extract valuable knowledge and regulate the memory capacity, but they can mutually improve the effectiveness of learning the knowledge for the given task in hand. Interestingly, a moderate level of forgetting may even improve the task performance rather than disadvantaging it. We suggest that the interplay between rapid memory formation, consolidation, and forgetting processes points to a practical and effective approach for learning agents to acquire and maintain useful knowledge from experiences in a scalable manner.

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: Artificial Intelligence—*Learning*

General Terms

Algorithms, Design, Experimentation, Theory

Keywords

Memory, Forgetting, Adaptive Resonance Theory

1. INTRODUCTION

Memory plays a key role in reasoning and decision making by providing past relevant episodes to improve learnt knowledge [10]. An agent with very limited or partial observability can construct a complete picture about its task environment by remembering all relevant information from its memory. Some agent architectures have incorporated declarative memory systems to support different aspects of the agent's performance. For example, [10] and [6] demonstrate the use of episodic memory to improve task performance and survivability of agents in simulated environments. [2, 8, 1, 7] also show how memory can improve the realism or human-likeliness of virtual agents. Most of these architectures consider declarative memory as a flexible information storage that can perfectly store and accurately retrieve information.

Although episodic memory can provide useful information about previous experiences at any moment of time, a significant amount of computational resources may still be needed to process specific items in memory to support reasoning and decision making. As the tasks and the environment become more complex, it is often impossible to make use of all the stored information necessary to make the right decision. For example, in a real-time environment, an agent may not have enough time to search and recall all relevant information to support its decision while it is performing some tasks. On the other hand, the agent may need to obtain enough information to acquire more compact, generalized, and efficient knowledge structure through a particular learning algorithm in order to make a timely and appropriate decision. Furthermore, most existing memory architectures for agents assume that all stored information is always relevant and consistent despite its limited capacity and possible erroneous inputs. The limitation of memory to keep all relevant and useful information is an important practical issue for learning agents but still seldom concerned.

According to neuropsychology, human memory systems have been known to be as non-unitary and partitioned into different types. Long-term declarative memory has been divided into episodic memory enabling one to remember personal experiences in specific manner and semantic memory that stores concepts, rules, and general facts [15]. It has also been considered that a consolidation process gradually transfers the specific items from episodic memory into general facts and rules in semantic memory [15, 3]. Episodic memory (particularly located in hippocampal area in the brain) and semantic memory (distributed across neo-cortical area in the brain) are anatomically separated but interconnected. It is suggested that these complementary memory systems prevent interferences of new information to old memories [9]. The consolidation between the two memory systems also implies that some forgetting processes may reg-

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

ulate and shape the memorized items to optimize performance as if the brain minds about the usefulness of memory traces rather than fidelity [12, 17]. It may be the case that perfect memory retrieval would burden the brain with too much details at the expense of remembering useful information.

In this paper, we propose a computational model of such a dual memory system based on a composition of fusion Adaptive Resonance Theory (ART) neural networks [14], which inherently serve learning, categorization, and recall operations. The dual memory model consists of an episodic memory that records agent's experience in real time and a semantic memory that captures factual knowledge relevant to the environment. The memory system is designed in such a way that the agent can store, recall, and playback experienced episodes in a sequential manner beyond individual momentary events. The availability of both the episodic and semantic memory modules enables a resource-bounded agent to focus on the situation and tasks in hand by capturing the episodic experience on-the-fly into a temporary memory store and deferring the resource-intensive learning of factual knowledge to a later stage through a memory consolidation process from episodic to semantic memory. In addition, the use of episodic memory as a buffer allows an agent to select and iterate through the relevant past cases as many times as needed at a later time, based on the updated awareness of the tasks and environment. This naturally leads to more effective learning comparing with learning the knowledge in an online manner while performing the task. To prevent the episodic memory from overloading, we further present a forgetting mechanism that associates each memory category with a time-decaying memory strength and removing unimportant traces or categories from the model.

The proposed dual-memory model has been evaluated using a real-time first-person-shooter video game called Unreal Tournament to support a non-player-character (NPC) agent to learn from experiences and improve performance. Surprisingly, we find that the memory model not only improves the task performance but in some cases, a moderate level of forgetting even results in more effective learning. Further examinations on the effects of forgetting show that selecting and pruning erroneous and outdated patterns promotes more efficient and robust learning.

The rest of the paper is organized as follows. Section 2 reviews related works in modelling declarative memory for autonomous agents and discusses some computational and neuropsychological accounts of consolidation and forgetting processes. The paper continues to describe the proposed memory model in Section 3. The section also formulates and analyses the characteristic of the memory model. Section 4 describes the implementation and the experiments of the memory system as parts of the non-player character agent in Unreal Tournament. The last section concludes the paper with some future directions.

2. RELATED WORK

Various types of declarative memory have been devised in recent years as parts of agent architectures. Most of them are developed as unitary systems, serving mainly the functionality of episodic memory to store linearly ordered traces of experiences or log records. This kind of sequential records has been demonstrated to optimize the agent's task performance [10] and improve the survivability of artificial life beings in virtual environment [6]. In the sequential traces model of episodic memory, specific mechanisms to search, retrieve, and recall the appropriate memory items are necessary to support performance. Each entry in episodic memory may also refer to other cognitive traits like procedural knowledge (in SOAR) [10] or emotions [1] to enhance the capability of memory operations and support complex behaviour.

To improve realism and interactivity, episodic memory entries can be associated with temporal relationships. For example, each entry can be associated with temporal weights to produce the effect of recency [8, 7] or generating temporal granurality [2]. This temporal association can emulate forgetting in which some details of memory item are suppressed according to time. However, these architectures still exclude memory consolidation processes to transfer and reorganize the contents of memory, nor consider the forgetting as beneficial to the overall performance.

On the other hand, it has been known that information memorized in the brain are subject to consolidation to make more general experience-independent forms of knowledge [3]. Memory traces from past experiences are played back before performing actions in similar or relevant contexts [11, 5]. Meanwhile, forgetting may take place removing or suppressing less useful memory items [17]. This forgetting dynamics of memory has been recognized as computationally beneficial to reduce inconsistency and the complexity of reasoning by discarding irrelevant information [16]. In multiagent systems, forgetting can also be useful for resolving conflicts between agents [4].

Our work in this paper also explores the role of episodic memory in agents. However, the main focus is to look at the memory consolidation process to extract useful knowledge and the discard of irrelevant information through forgetting. Instead of just applying episodic memory as a unitary flexible storage system, we make dual episodic-semantic memory systems working together to acquire useful knowledge through the interplay of consolidation and forgetting processes.

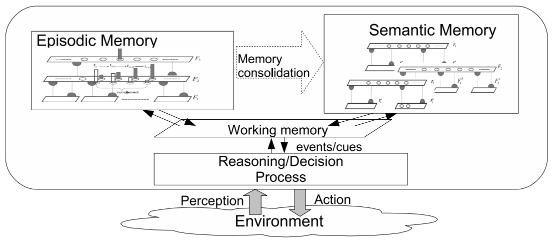


Figure 1: The Episodic-Semantic Memory

3. THE DUAL MEMORY MODEL

The proposed memory model is considered as a part of the reasoning system of an agent architecture (Figure 1). It can be assumed that at each point in time, a snapshot of an agent's situation and perception can be encoded as an individual event and held temporarily in working memory. Episodic memory automatically stores and organizes the events in a sequential order into cognitive units of episodes, which are then periodically transferred to semantic memory.

In general, the memory system goes through different stages of operation as follows:

- Events captured in working memory are continuously stored in episodic memory.
- Periodically, traces in episodic memory are readout to working memory triggering learning in semantic memory.
- Memory items in episodic and semantic memory can any time be recalled based on certain memory cues through a process of pattern completion or reconstruction.
- Rarely accessed items in episodic memory will be removed (forgotten).

The remaining parts of this section first formulate the notion of events and episodes before presenting the details of the model and process as follows.

3.1 Events and Episodes

An *event* is a snapshot of perceived experience at one moment in time which can be defined as a collection or a tuple of attributes.

Definition 1. An event ε is a tuple reflecting a moment of experience such that $\varepsilon = (\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k)$. Each attribute $\mathbf{v}_i$ is defined as a tuple such that $\mathbf{v}_i = (v_1^i, v_2^i, .., v_l^i)$ and v_j^i is a normalized real value $v_j^i \in [0, 1]$.

By normalizing the value of v_j^i, an event can represent a proposition with a binary truth at the extreme values (0 or 1) or a certain degree in between (fuzzy values). To recall a stored event, an *event cue* can be expressed with the same tuple structure. However, some elements of the cue may be left unspecified and the recall operation would reconstruct the target event. Figure 2(a) shows an example of an event representation in the Unreal Tournament domain used in our experiment (explained later in detail).

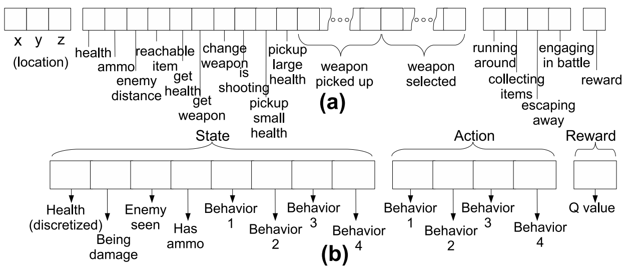


Figure 2: (a) Event Encoding; (b) Input (Output) representation of *RLBot* NPC in the experiment

An *episode*, on the other hand, can be defined as a finite list of events collected in a temporal order.

Definition 2. An episode E is a sequence of events such that $\mathrm{E} = [\varepsilon_{t_0}, \varepsilon_{t_1}, ..., \varepsilon_{t_n}]$, where t_j denotes the relative time point wherein the event ε_{t_j} occurs.

In contrast to the event structure, which always has the same length of tuple, the length of the episode sequence may vary.

3.2 Building Blocks

The proposed memory model is based on fusion Adaptive Resonance Theory (ART) [14], which applies a myriad of learning paradigms to recognize and learn an incoming stream of input patterns across multiple channels in real time. It employs a bi-directional process of categorization and prediction to find the best matching category (resonance). It also learns continuously by updating the weights of neural connections at the end of each search cycle. ART may also grow dynamically by allocating a new category if no match can be found. This type of neural network is chosen as the building block of our memory model as it enables continuous formation of memory with adjustable vigilance of categorization to control the growth of the network and the level of generalization. Specifically, a fusion ART model can be defined as follows:

Definition 3. Suppose F_1^k and F_2 are the kth input (output) field and the category field of fusion ART (Figure 3) respectively for $k = 1, .., n$. Let $\mathbf{x}^k$ denote the F_1^k activity vector and $\mathbf{w}_j^k$ denote the weight vector associating kth field with jth node in F_2. F_1^k is associated with choice parameter $\alpha^k > 0$, learning rate $\beta^k \in [0, 1]$, contribution parameter $\gamma^k \in [0, 1]$, and vigilance parameter $\rho^k \in [0, 1]$.

Definition 4. Choice function T_j returns the activation value of category j such that:

$$T_j = \sum_{k=1}^{n} \gamma^k \frac{|\mathbf{x}^k \wedge \mathbf{w}_j^k|}{\alpha^k + |\mathbf{w}_j^k|} \quad (1)$$

where the fuzzy AND operation $\wedge$ is defined by $(\mathbf{p} \wedge \mathbf{q})_i \equiv \min(p_i, q_i)$, and the norm $|.|$ is defined by $|\mathbf{p}| \equiv \sum_i p_i$ for vectors $\mathbf{p}$ and $\mathbf{q}$.

Definition 5. Template matching m_j^k is the matching value or similarity of category j with the input $\mathbf{x}^k$ such that:

$$m_j^k = \frac{|\mathbf{x}^k \wedge \mathbf{w}_j^k|}{|\mathbf{x}^k|} \quad (2)$$

A category J of F_2 field is in *resonance condition* if and only if:

$$T_J = \max\left\{T_j : \forall k, m_j^k \geq \rho^k, \text{for all } F_2 \text{ category } j\right\} \quad (3)$$

Definition 6. Given the selected category J, *Template learning* modifies the weights associated with J such that:

$$\mathbf{w}_J^{k(\text{new})} = (1 - \beta^k)\mathbf{w}_J^{k(\text{old})} + \beta^k(\mathbf{x}^k \wedge \mathbf{w}_J^{k(\text{old})}) \quad (4)$$

The corresponding weight vector of the chosen F_2 node J can be readout into the input field F_1^k such that $\mathbf{x}^{k(\text{new})} = \mathbf{w}_J^k$.

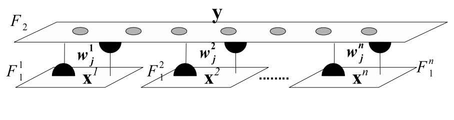


Figure 3: Fusion ART Neural Network

If no existing F_2 category can be found in resonance condition with the current input, a new category is recruited to represent the current input pattern. This implies that the ART network can grow to accommodate the incoming stream of different input patterns. The growth rate of the categories depends on how much the incoming patterns differ from one another and is adjustable through adjusting the vigilance parameters (ρ^k). Lower vigilance may tolerate differences more than the higher one and hence lead to a slower growth.

3.3 Episodic Memory

From *Definition* 1, it is clear that an event corresponds directly to the input vector representation in the fusion ART model. Specifically, an event ε^k is encoded into an input vector $\mathbf{x}^k$ to be learnt by fusion ART. On the other hand, based on *Definition* 2 an episode corresponds to the sequence of selected categories (events) collected in a temporal order. An episodic memory model can be built to store events and episodes by combining two fusion ARTs: one for storing events and the other for episodes. Figure 4 shows that events are represented and processed by the ART network between F_1^k and F_2 fields. In this case, an event can be learnt and retrieved by selecting a matching category in F_2 and readout the pattern back to F_1^k. In a word, each F_2 category j represents a single event.

To capture episodes, another layer of ART gets input from categories selected in F_2 from the other network. Let $\mathbf{y} = (y_0, y_1, .., y_m)$ denote the F_2 activity vector. If J is the F_2 category currently selected by the resonance search, then $y_J = 1$ and $y_j^{(new)} = y_j^{(old)}(1-\tau)$ for all F_2 category $j \neq J$ and $\tau \in (0,1)$.

Lemma 1. *Consider the F_2 activity vector $\mathbf{y}$ as described above. If $y_J = 1$ for J is the currently selected category and $y_j^{(new)} = y_j^{(old)}(1-\tau)$ for all $j \neq J$ with $\tau \in (0,1)$, then $\mathbf{y}$ reflects the relative order of category selection in F_2 such that $y_{j_t} > y_{j_{t-1}} > y_{j_{t-2}} > ..y_{j_{t-m}}$ for j_t is the node selected at relative time point t.*

Proof. Given $0 < (1-\tau) < 1$, it is clear that $1 > (1-\tau) > (1-\tau)^2 > .. > (1-\tau)^m$. Since $y_{j_t} = 1$ for any category j_t and from $y_j^{(new)} = y_j^{(old)}(1-\tau)$ it follows that $y_{j_{t-m}} = (1-\tau)^m$, consequently $y_{j_t} > y_{j_{t-1}} > y_{j_{t-2}} > .. > y_{j_{t-m}}$. □

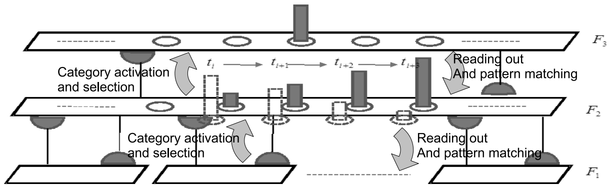
The list of events with their relative order expressed as $\mathbf{y}$ vector becomes the input to the upper ART network (between F_2 and F_3 fields) to be learnt as an episode. Based on the bi-directional activation and matching process in ART, a category I representing an episode is selected in F_3 such that

$$T_i = \frac{|\mathbf{y} \wedge \mathbf{w}_i|}{|\mathbf{w}_i|},\, m_i = \frac{|\mathbf{y} \wedge \mathbf{w}_i|}{|\mathbf{y}|}, \text{and}$$

$$T_I = \max\{T_i,\, m_i \geq \rho_2, \text{for all}\, F_3\, \text{node}\, i\}.$$

Parameters α, γ, and the field index k are omitted as the upper network only has a single input field (F_2). Given the selected category I, learning takes place such that $\mathbf{w}_I^{(\text{new})} = (1-\beta_2)\mathbf{w}_J^{(\text{old})} + \beta_2(\mathbf{y} \wedge \mathbf{w}_I^{(\text{old})})$. ρ_2 and β_2 are the vigilance and learning rate parameters respectively of the field F_2.

If consecutive events are received in F_1^k, a pattern of their sequence can be formed in F_2 as vector $\mathbf{y}$ and can be used as an input to select category I in F_3. In this case, the input patterns act as memory cues. To reproduce the original sequence of events of the episode, two stages of readout



Category activation and selection
Reading out And pattern matching
Category activation and selection
Reading out And pattern matching
t_i
t_{i+1}
t_{i+2}
t_{i+3}
F_3
F_2
F_1

Figure 4: The Episodic Memory Model

operation are conducted by firstly reading out the sequential pattern of the episode into vector $\mathbf{y}$ and secondly an F_2 category J is selected such that $T_J = \max(\overline{y}_j : \overline{y}_j = 1 - y_j, \text{for all}\, F_2\, \text{category}\, j)$ and readout to the corresponding F_1^k before reset to zero or $y_J = 0$. The readout cycles continue until $\mathbf{y} = \mathbf{0}$.

3.4 Semantic Memory

Different from episodic memory, we view that the semantic memory is not unitary, with different fusion ARTs representing different structure of knowledge. In contrast to episodic memory, each entry in semantic memory generalizes similar inputs into the same category rather than as separate entries. Each input field of a semantic memory represents a property or an attribute of a concept. The generalization can be achieved by lowering the vigilance parameter ρ^k so that slightly different input patterns will still activate the same category. The value of an attribute can be paired, as described below, so that the ART learning can generalize the value as a range of values.

Let $\mathbf{I}^k$ be the input vector for $F_1^k, I_i^k \in [0,1]$. $\mathbf{I}^k$ is augmented with $\bar{\mathbf{I}}^k$ such that $\bar{I}_i^k = 1 - I_i^k$. The activity vector $\mathbf{x}^k$ of F_1^k thus augments the input vector $\mathbf{I}^k$ with its complement $\bar{\mathbf{I}}^k$ which are learnt as a $\mathbf{w}_j^k$. Let $(w_{ij}^k, \overline{w}_{ij}^k)$ be the corresponding pair of $\mathbf{w}_j^k$. The value of the connection becomes less specified when $w_{ij}^k \neq 1 - \overline{w}_{ij}^k$.

Lemma 2. *For the pair $(w_{ij}^k, \overline{w}_{ij}^k)$ of $\mathbf{w}_j^k$ described above, if $w_{ij}^k \neq 1 - \overline{w}_{ij}^k$, any corresponding complemented input pair $(I_{ij}^k, \overline{I}_{ij}^k)$ will have a maximum matching value or always in resonance as long as $w_{ij}^k \geq I_{ij}^k \geq 1 - \overline{w}_{ij}^k$.*

Proof. Let the pair $(x_{ij}^k, \overline{x}_{ij}^k) \equiv (I_{ij}^k, \overline{I}_{ij}^k)$. It is clear that if $x_{ij}^k \leq 1 - \overline{w}_{ij}^k$ then $\overline{x}_{ij}^k \leq \overline{w}_{ij}^k$. Thus, $(x_{ij}^k, \overline{x}_{ij}^k) \wedge (w_{ij}^k, \overline{w}_{ij}^k) = (x_{ij}^k, \overline{x}_{ij}^k)$ such that *template matching* $m_j^k = \frac{|\mathbf{x}^k \wedge \mathbf{w}_j^k|}{|\mathbf{x}^k|} = \frac{|\mathbf{x}^k}{|\mathbf{x}^k|} = 1$ □

It can be considered that a stored value is unspecified if the values of the corresponding complementary pair $(w_{ij}^k, \overline{w}_{ij}^k)$ are equal.

Similar to episodic memory, the content of semantic memory can be retrieved by pattern completion based on memory cues. Figure 5 illustrates various types of semantic memory. A single fusion ART may consist of domain specific associative rules (e.g a set of association between a certain object and its location in the environment, a set of rule associating the effectiveness of a certain weapon and the distance to the opponent) or generic causal relations associating a particular type of event to another that follows. These types of semantic knowledge can be derived by exposing the played back items from the episodic memory to the input of the semantic memory using a lower vigilance parameter ρ^k and a smaller learning rate β^k such that similar instances may gradually be clustered together regardless of their order.

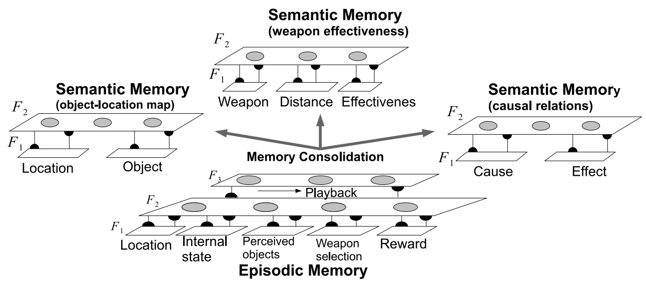


Figure 5: Different types of Semantic Memory and the Memory Consolidation Process

3.5 Memory Consolidation

Knowledge can be transferred from episodic memory by playing back stored episodes or reading out category nodes in F_3 field to each corresponding input fields. The readout events are then passed to the working memory, one at a time, to be shared by other memory modules. Depending on the kind of semantic structure and the domain problem, different subsets of items in working memory are connected to different input fields in semantic memory.

Definition 7. Let $\mathbf{O}$ be the vectors in F_1 fields read out from the selected category in episodic memory as a result of the playback process. Function $\mathcal{M} : \mathcal{P} \rightarrow \Theta$ maps the read out vectors from episodic memory into the corresponding input vectors of semantic memory, wherein $\mathcal{P}$ and Θ are the vector space of the input fields of episodic memory and semantic memory respectively.

During the episodic memory playback, $\mathbf{x}' = \mathcal{M}(\mathbf{O})$ becomes the input vector to learn in semantic memory. The played back vectors can be the results of the top-down readout process or it can be the results of a retrieval operation based on some memory cues. If $\mathbf{x}' = \mathcal{M}(\varepsilon)$, it can be said that semantic memory works standalone by learning directly from the received events ε bypassing episodic memory. A standalone version of semantic memory is also presented in this paper as one configuration to compare with the dual memory model in our experiment (explained later).

3.6 Forgetting

Forgetting is a mechanism to free up memory space in episodic memory by removing unnecessary items. Intuitively, a memory item can be considered unnecessary or obsolete if it is rarely accessed or recalled for long. The forgetting mechanism in the proposed model is applied to episodic memory for both the event layer (F_2) and the episode layer (F_3).

Definition 8. Given a category j (representing either an event or an episode) in a fusion ART structure, a memory strength $\mathcal{S}_j^t$ reflects how often category j is selected such that:

$$\mathcal{S}_j^t = \begin{cases} \mathcal{S}_j^{t-1} + (1-\mathcal{S}_j^{t-1})r_s & \text{if } j \text{ is selected at time t} \\ \mathcal{S}_j^{t-1}(1-\delta_s) & \text{otherwise} \end{cases} \tag{5}$$

where $r_s \in [0,1]$ and $\delta_s \in [0,1]$ are reinforcement and decay rate parameters respectively. A category j with $\mathcal{S}_j^t < \theta_s$ will be removed or pruned from memory including all the associated connections. If j is a new allocated category, it is assigned with an initial strength $\mathcal{S}_j^{init}$.

4. CASE STUDY

The episodic-semantic memory system is implemented and embedded into an autonomous non-player character (NPC) in a first-person-shooter video game called Unreal Tournament (UT). Our objectives are to test if the proposed memory model can produce useful knowledge for the agent and whether the forgetting process may sacrifice the agent's performance. The scenario of the game used in the experiment is "Deathmatch". The objective of each agent is to kill as many opponents as possible and to avoid being killed by others. In the game, two (or more) NPCs are running around and shooting each other. They can collect objects in the environment, like health or medical kit to increase its strength and different types of weapon and ammunition for shooting. The battle simulation in UT game is a suitable platform for evaluating memory tasks. Besides complex spatial maps and terrains, different objects and situations in the game may have some intricate relationships that should be memorized and remembered in non-trivial ways.

4.1 Weapon Learning Task

In the first experiment, we task the agent to learn the relationship between the type of weapon and its effectiveness to kill given the distance of the opponent agent. We compare the performance of different agents with reinforcement learning and the dual episodic-semantic memory.

4.1.1 The Baseline Agents for Comparison

All agents that we evaluate in the experiment play against an NPC agent called *AdvanceBot* that behaves according to hardcoded rules. There are four different hardcoded behavior modes in *AdvanceBot*: (1) Running around behaviour, in which the agent runs around exploring the environment randomly; (2) Collecting items behavior, in which the agent goes around and picks up collectible items; (3) Escaping from the battle situation, in which the agent turns and runs away from the opponent; (4) Engaging in battle, in which the agent approaches its opponent and shoots to kill it. *AdvanceBot* always chooses one of the four behaviors based on a set of predefined rules.

Under the battle engagement behavior, the agent also always tries to select the best weapon available for shooting. The weapon selection rules are based on some heuristics optimized for a certain environment map used in the game.

As a performance comparison, another agent (named *RL-Bot*) is made to employ the same set of behaviors but its selection is conducted dynamically based on a fusion ART neural network conducting reinforcement learning algorithm. The state, action, and reward vectors in Figure 2(b) correspond to the input fields in a fusion ART network of *RL-Bot*. The behavior pattern in the state vector represents the behavior (1 to 4) currently selected. The action vector indicates the next behavior to be selected. Based on the state field input and the reward cue (set to the maximum), the network searches the best match category node and reads out the output to the action field indicating the behavior type to be selected. The network then receives feedbacks in terms of the new state and reward (if any). The network learns by updating the weighted connections according to the feedback received and applying temporal difference methods [13] to update the reward field if the immediate reward is absent. The agent receives the reward signal (positive or negative) whenever it kills or is killed by another agent. In contrast to *AdvanceBot*, *RLBot* chooses an available weapon randomly in the battle engagement behavior. Another agent called *RLBot++* is also used to employ the same reinforcement learning model as *RLBot* but select the weapon based on the optimized predefined rules just like in *AdvanceBot*.

4.1.2 Episodic-Semantic Memory Based Agent

The proposed model is embedded in an agent with the same architecture as *RLBot*, but with the episodic and semantic memory modules running concurrently. The episodic memory captures episodes based on the event information in the working memory. An event from the UT game is encoded as a vector shown in Figure 2(a). There are four input fields in episodic memory for location, state, selected behavior, and the reward received. In the experiment, the vigilance of all input fields (ρ_e) and the F_2 field (ρ_s) are set to 1.0 and 0.9 respectively so that it tends to always store distinct events and episodes in response to the incoming events. At a certain period of time, the contents of the episodic memory is played back by reading out the events to the working memory. The reinstatement occurs in the period between different battles wherein one agent has just been killed and started to respawn in another place. The semantic memory then acquires the knowledge by learning from the recalled events. In the experiment, only one type of semantic memory about weapon effectiveness is learnt given the distance towards the enemy. Whenever the value of the reward field in the event vector is large enough to be considered as a successful killing (0.5 is the threshold), the values of weapon selected, opponent distance, and reward (or the effectiveness to kill) fields are fed and learnt by the semantic memory.

Figure 5 shows the fusion ART network of the semantic memory for weapon effectiveness used in the experiment. The network has three input fields: the Weapon field representing the identity of the weapon (F_1^a); the Distance field representing the distance between the agent and its opponent at the time of shooting (F_1^b); and the Effectiveness field representing the chance to kill the enemy (F_1^c). In the experiment, the vigilance of the Weapon (ρ^a), Distance (ρ^b), and Effectiveness (ρ^c) fields are 1.0, 0.9, and 0.8 respectively. The learning rate β^a, β^b, and β^c are 1.0, 0.1, and 0.2 respectively. The agent reasoning system can use the knowledge in the semantic memory by providing the current distance to the opponent while setting up the effectiveness to maximum (the greatest chance of killing) as memory cues. The retrieved values support the agent to decide which weapon to select during the battle. If the cue is not recognized, a random weapon is selected.

As a comparison, we also implement an agent with a standalone version of semantic memory as mentioned in the previous section. The agent, called *AssocBot*, learn the weapon selection knowledge by directly associating weapon and enemy distance from the incoming events without consolidating episodic memory or $\mathbf{x}' = \mathcal{M}(\varepsilon)$. This direct semantic memory only learn the event whenever the NPC shot hits

Table 1: Sample Rules Learnt in Semantic Memory

```
IF distance is not so far [1800 2099]
  THEN ASSAULT_RIFLE effectiveness 0.07
IF distance is very near [300 599]
  THEN ASSAULT_RIFLE effectiveness 0.048
IF distance is extremely near [0 299]
  THEN SHOCK_RIFLE effectiveness 0.946
IF distance is very near [300 599]
  THEN ROCKET_LAUNCHER effectiveness 0.932
```
```
weapon range categorization: extremely near:0-299;
very near:300-599;near:600-899;medium near:900-1199;
not so near:1200-1499;midrange:1500-1799;not so far:1800-2099;
medium far:2100-2399;far:2400-2699;very far:2700-2999;
exremely far:3000 or more
```

the opponent.

Table 1 illustrates sample learnt rules of weapon effectiveness translated into symbolic forms. Each rule corresponds to a category node in F_2 layer of the semantic memory. The generalization employed using Fuzzy operators makes it possible to represent the rule with a range of values like the rule antecedents shown above. Table 1 also shows the symbolic categorization of the distance range for interpreting the rules. The experiment also uses forgetting in episodic memory with $\mathcal{S}_j^{init}$, threshold (θ_s), and reinforcement rate (r_s) set to 0.5, 0.0001, and 0.5 respectively. To evaluate the effect of forgetting, different decay rates (δ_s) in the events field F_2 are used: 0 (no forgetting), 0.005, 0.01, and 0.02.

4.1.3 Results

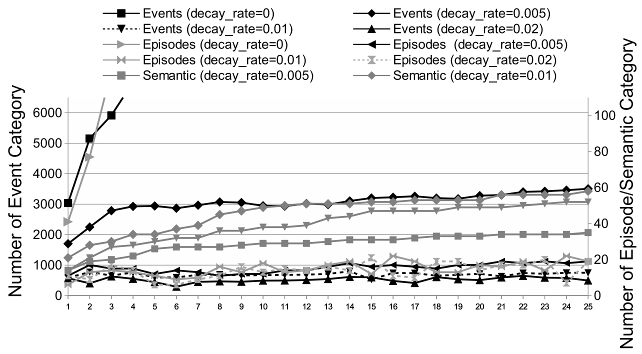


Figure 6: Memory usage for events, episodes, and transferred semantic knowledge with different forgetting decay rate during the game play

Experiments are conducted by letting *RLBot*, *RLBot++* and the memory-based *RLBot* (called *MemBot*) with different forgetting decay rates (δ_s=0.005, δ_s=0.01, and δ_s=0.02) to individually play against *AdvanceBot*. In addition, the direct semantic-memory-based *RLBot* (*AssocBot*) is also put to the test. For practical reason, *MemBot* without forgetting (δ_s=0) is excluded from performance comparison as the program overloads the system memory soon after the game starts causing the system to halt and the agent refrains from playing. A single experiment run consists of 25 games or trials, which is counted whenever the agent kills or is killed by another agent.

Figure 6 shows the memory size taken up in the episodic memory (in terms of the number of nodes in F_2 and F_3 of a *MemBot*) and the number of nodes created in the semantic memory with different δ_s in F_2 sampled from a single run against *AdvanceBot*. Without forgetting ($\delta_s = 0$), the memory space is taken up rapidly into its limit after about three trials. In contrast, the forgetting mechanism can make the memory size converge and stabilize at certain points. Hence the agent can always perform and learn continuously. It is clearly shown that the larger the decay rate, the smaller number of categories is produced in episodic memory. Interestingly, a low semantic memory decay rate (e.g δ_s=0.005) creates lesser categories comparing with those obtained with higher rates (e.g δ_s=0.01 and δ_s=0.02).

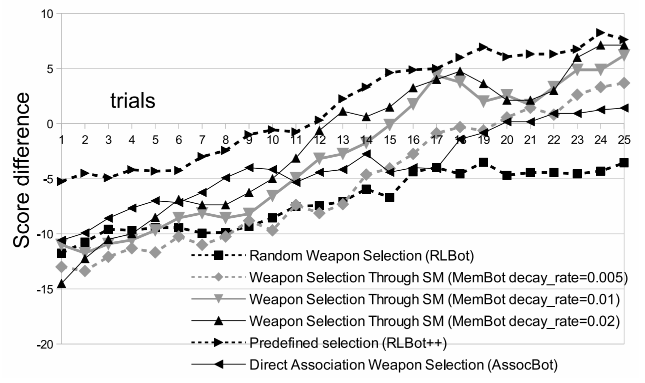


Figure 7: Performance of *RLBot*, *RLBot++*, *MemBot*, *AssocBot* over 25 trials

Figure 7 plots the performance of *RLBot*, *RLBot++*, *MemBot*, *AssocBot* with different δ_s in terms of game score differences against *AdvanceBot* averaged over four independent runs. It shows that incorporating the proposed episodic and semantic memory model improves the learning which results in a much better performance than using the reinforcement learning alone (with random weapon selection). This indicates that the semantic memory can learn useful knowledge about weapon selection. It is also shown that although *AssocBot* can learn and improve better than RLBot, it is still marginally inferior than MemBot which applies consolidation but with the smallest forgetting decay rate ($\delta_s = 0.005$). It indicates that the consolidation can be advantageous for a learning agent. The reason could be that, with consolidation, the same categories might be activated or selected in semantic more than once as the result of the playing back episodic memory. This reactivation of certain categories shapes and reinforces knowledge in semantic memory, whereas learning the semantic directly without consolidation may produce over-generalization.

Surprisingly, the results also indicate that with a higher forgetting rate (e.g δ_s=0.01 and δ_s=0.02), the performance and learning efficiency of MemBot are better than those obtained with the smaller one (δ_s=0.005) and can eventually reach the performance using the optimized rules model. In other words, forgetting less important things faster can make learning better. One explanation of this beneficial effect of forgetting is that events in the UT game related to weapon use are noisy and full of inconsistencies. Thanks to forgetting, events that could impair the consolidated knowledge are filtered out before being generalized in semantic memory. The semantic memory would thus end up with the appropriate generalization and some specific but necessary information.

4.2 Noisy Event Recognition Task

To validate our explanation about the significance of forgetting in noisy and inconsistent environment, we conduct another test to see how forgetting contributes to the retrieval accuracy in a situation where some noise distribution is introduced to the input events. The test is conducted off-line based on the recorded events of selected UT game sessions. The forgetting test is applied to episodic memory only.

4.2.1 Testing Configuration

The episodic memory model described above is made to learn from different sets of events recorded from a UT game session with the same event structure applied in the above experiment. The original set of events consists of 77350 events and 1000 game sessions. We assume that, the memory learns a sequence of events as a single episode at the end of each session. To simulate the noisy environment, different sets of events are generated by introducing two different rates of noises following a Gaussian distribution. Two sets of events are used with 5% and 10% noise rates.

Episodic memory learns each noisy data set to generate the representation of events and episodes in memory. The evaluation is conducted to obtain the retrieval accuracy by measuring the difference of the selected episodes with the ones selected by episodic memory that has learnt the original (without noise) data set. It should be noted that the accuracy is not measured based on recall or reconstruction of the original episodes but only based on the recognition of the presented episodes.

The retrieval cues used in the evaluation are one-fifth portions of the target episode taken from the original data set (without noise). We compare the retrieval accuracy of both episodic memory configurations that learn different data sets with and without forgetting. The parameters of event level forgetting used in the experiment are as follow: initial confidence $\mathcal{S}_j^{init} = 0.5$, decay factor $\delta_s = 10^{-4}$, reinforcement parameter $r_s = 0.5$, and threshold $\theta_s = 0.1$. For episode level forgetting: the parameter values are initial confidence $\mathcal{S}_j^{init} = 0.5$, decay factor $\delta_s = 0.008$, reinforcement parameter $r_s = 0.5$, and threshold $\theta_s = 0.1$. The vigilance parameters ρ for events and episodes level are 0.5 and 0.95 respectively.

4.2.2 Results

Figure 8 clearly shows that, in a noisy condition, forgetting helps episodic memory to retrieve the correct episodes despite some learnt events and episodes have been discarded from memory. In both 5% and 10% level of noises, the performance with forgetting is always superior to the configurations without forgetting. The results support our hypothesis that forgetting can contribute to the overall performance of the agent when information perceived from the environment are noisy and full of inconsistencies.

5. CONCLUSION

We have presented an explicit dual memory model for agents by integrating two separate modules of episodic and semantic memory. The stored contents of episodic memory can be recalled to derive abstract knowledge and general

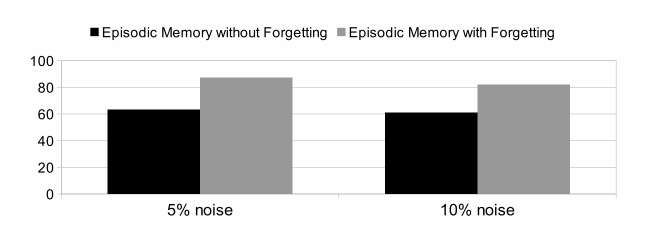


Figure 8: Accuracies in retrieval with and without forgetting in noisy environment

facts, that in turn are transferred to more permanent forms in semantic memory. The episodic and semantic memory modules are realized with fusion ART (adaptive resonance theory) neural networks as two independent but connected networks operating in different paces of learning. In line with theories and findings in neuropsychology, the asymmetrical rate of learning with some periodical consolidation between the two enables the acquisition of useful knowledge without risking to loose prior entries. A forgetting mechanism is also applied to regulate the size of memory by removing insignificant entries.

Our experiments confirm that an explicit episodic-semantic memory model can improve the agent learning and performance by acquiring useful knowledge for the task at hand through memory consolidation, relieving the agent from continuously reasoning and processing the information for learning. It is also demonstrated that the forgetting regulates the memory size while the performance is still improving. Moreover, the experiment shows faster forgetting can result in better learning. This indicates that the forgetting can successfully filter insignificant entries while maintaining the useful ones. The findings can inspire the exploration of forgetting as a useful feature of intelligent agents and machine learning systems in general.

In the future, we shall extend our model to learn more useful and general purpose semantic structures from episodic memory. We shall also extend our study to look at how episodic memory model may contribute directly to the performance of the agent rather than just as a transient structure. The forgetting mechanism can also be extended by applying different variations of memory strength functions to include task-related aspects like rewards, risks, or penalties. This may reveal the potential of the dual episodic-semantic model as effective memory systems that continuously and mutually process, learn, and forget information.

6. ACKNOWLEDGEMENTS

This research is supported by the DSO National Laboratories under research grant DSOCL11258.

7. REFERENCES

[1] C. Becker-Asano and I. Wachsmuth. Affective computing with primary and secondary emotions in a virtual human. *Autonomous Agents and Multi-Agent Systems*, 20(1):32–49, 2010.

[2] C. Brom, T. Korenko, and J. Lukavsky. How do place and objects combine? what-where? memory for human-like agents. *Intelligent Virtual Agents*, LNCS 5773, pages 42–48. Springer Berlin, 2009.

[3] M. A. Conway, J. M. Gardiner, T. J. Perfect, S. J. Anderson, and G. M. Cohen. Changes in memory awareness during learning: The acquisition of knowledge by psychology undergraduates. *Journal of Experimental Psychology*, 126(4):393–413, 1997.

[4] E. Erdem and F. Paolo. Forgetting actions in domain descriptions. In *Proceedings of the 22nd national conference on Artificial Intelligence (AAAI 2007)*, volume 1, pages 409–414. AAAI Press, 2007.

[5] A. S. Gupta, M. A. A. van der Meer, D. S. Touretzky, and A. D. Redish. Hippocampal replay is not a simple function of experience. *Neuron*, 65(5):695–705, 2010.

[6] W. C. Ho, K. Dautenhahn, and C. L. Nehaniv. Computational memory architectures for autobiographic agents interacting in a complex virtual environment: A working model. *Connection Science*, 20(1):21–65, 2008.

[7] Z. Kasap, M. B. Moussa, P. Chaudhuri, and N. Magnenat-Thalmann. Making them remember–emotional virtual characters with memory. *IEEE Computer Graphics and Applications*, 29(2):20–29, 2009.

[8] M. Y. Lim, R. Aylett, W. C. Ho, S. Enz, and P. Vargas. A socially-aware memory for companion agents. *Intelligent Virtual Agents*, LNCS 5773, pages 20–26. Springer Berlin, 2009.

[9] J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning memory. *Psychological Review*, 102(3):419–457, 1995.

[10] A. M. Nuxoll and J. E. Laird. Extending cognitive architecture with episodic memory. In *Proceedings of the 22nd national conference on Artificial Intelligence (AAAI 2007)* Volume 2, pages 1560–1565. AAAI Press, 2007.

[11] C. Ranganath, A. P. Yonelinas, M. X. Cohen, C. J. Dy, S. M. Tom, and M. D'Esposito. Dissociable correlates of recollection and familiarity within the medial temporal lobes. *Neuropsychologia*, 42(1):2–13, 2004.

[12] A. J. Silva and S. A. Josselyn. Cognitive neuroscience: The molecule of forgetfulness. *Nature*, 418:929–930, 2002.

[13] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, 1998.

[14] A.-H. Tan, G. A. Carpenter, and S. Grossberg. Intelligence Through Interaction: Towards A Unified Theory for Learning. In *International Symposium on Neural Networks (ISNN 2007)*, LNCS 4491, pages 1098–1107, Springer Berlin 2007.

[15] E. Tulving. *Elements of episodic memory.* Oxford University Press, 1983.

[16] K. Wang, A. Sattar, and K. Su. A theory of forgetting in logic programming. In *Proceedings of the 20th national conference on Artificial intelligence (AAAI 2005)*, volume 2, pages 682–687. AAAI Press, 2005.

[17] J. T. Wixted. The psychology and neuroscience of forgetting. *Annual Review of Psychology*, 55:235–269, 2004.

Improved Use of Partial Policies for Identifying Behavioral Equivalence

Yifeng Zeng
Dept. of Computer Science Dept. of Automation
Aalborg University Xiamen University
Aalborg, Denmark Xiamen, China
yfzeng@cs.aau.edu yfzeng@xmu.edu.cn

Yinghui Pan
Dept. of Automation Information Management
Xiamen University Jiangxi Univ. of Fin.& Eco.
Xiamen, China Jiangxi, China
pyhui@xmu.edu.cn

Hua Mao
Dept. of Computer Science
Aalborg University
Aalborg, Denmark
huamao@cs.aau.edu

Jian Luo
Dept. of Automation
Xiamen University
Xiamen, China
jianluo@xmu.edu.cn

ABSTRACT

Interactive multiagent decision making often requires to predict actions of other agents by solving their behavioral models from the perspective of the modeling agent. Unfortunately, the general space of models in the absence of constraining assumptions tends to be very large thereby making multiagent decision making intractable. One approach that can reduce the model space is to cluster *behaviorally equivalent* models that exhibit identical policies over the whole planning horizon. Currently, the state of the art on identifying equivalence of behavioral models compares partial policy trees instead of entire trees. In this paper, we further improve the use of partial trees for the identification purpose and develop an incremental comparison strategy in order to efficiently ascertain the model equivalence. We investigate the improved approach in a well-defined probabilistic graphical model for sequential multiagent decision making - interactive dynamic influence diagrams, and evaluate its performance over multiple problem domains.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms, Experimentation

Keywords

decision making, agent modeling, behavioral equivalence

1. INTRODUCTION

Decision making in interactive multiagent settings becomes complicated mainly due to unknown actions of other agents from the eyes of the modeling agent. A general solution is to model other agents using a specific representation and then solve the models to predict their actions. Unfortunately, the model space ascribed to other agents is often very large thereby making multiagent decision making intractable. A line of research has exploited the concept of *behavioral equivalence (BE)* to reduce the dimensionality of the model space [2, 12, 13]. A pair of models are behaviorally equivalent if the models have identical solutions that are normally represented as policy trees. We may consider to group a set of BE models and choose a representative model for each cluster. Clustering BE models to reduce the model space will not compromise the solution optimality since it is the prescriptive aspects of the models and not the descriptive that matter to the modeling agent. Recently, a well-defined probabilistic decision making framework - interactive dynamic influence diagram (I-DID) [6] - has intensively exploited BE models for achieving the solution scalability.

I-DIDs are probabilistic graphical models for sequential decision making in uncertain multiagent settings. They generalize dynamic influence diagrams (DIDs) [14] to multiagent settings analogously to the way that interactive partially observable Markov decision processes (I-POMDPs) [9] generalize POMDPs. As we may expect, solving I-DIDs is computationally very hard. This is because the state space in I-DIDs includes the models of other agents in addition to the traditional physical states. As the agents act, observe and update beliefs, I-DIDs must track the evolution of the models over time. The exponential growth in the number of models over time also further contributes to the dimensionality of the state space. This is further complicated by the nested nature of the state space.

Previous I-DID solutions, including both exact and approximate ones, mainly exploit the concept of BE to reduce the dimensionality of the state space. For example, the proposed technique in [5] updates only those models that lead to behaviorally distinct models at the next time step. It results in a minimal model space. A central component of this technique is the way of identifying equivalence of behavioral models ascribed to other agents. It firstly builds policy trees for the associated models and then checks the equality of every path in the entire trees. Since the size of the policy

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

tree increases exponentially as the horizon increases, the BE identification method becomes computationally intractable in the case of large horizons. Additionally, based on this identification technique, the current I-DID solution does not scale desirably to large horizons because it groups only exact BE models thereby still resulting in a large model space. One leading solution to further reduce the model space is to cluster models that are approximately BE.

Recently, one efficient way of identifying approximately BE between models is to compare their partial policy trees instead of entire ones [19]. The depth of the partial trees is determined by a given approximate measure of BE. This defines an approximately BE that could group more models together resulting in less numbers of BE classes. However, the proposed method still requires an expansion of a full size of the partial policy tree that has a symmetric structure with a uniform length on all paths. This may lead to a strict condition on approximating BE while the identification using partial trees is not executed efficiently.

In this paper, we present an improved version of using partial trees to identify approximately BE models. We make a general definition on a partial policy tree that allows different lengths for its paths. The maximum path length is calculated according to a predefined value on measuring the approximation between two BE models. The measurement value quantifies the allowed divergence between updated beliefs in the policy trees. To efficiently use partial policy trees to determine approximately BE, we propose an incremental identification approach: we expand the trees only when comparing the updated beliefs at the leaf nodes is not sufficient to ascertain the model equivalence. The comparison expects to be terminated before it reaches the maximum length for all paths. By doing this we maintain a rather small set of policy paths instead of all full paths in the partial trees. Specifically, the incremental method is applicable even when the maximum length can't be computed in some problem domains.

Furthermore, we may group more approximately BE models by comparing only a subset of policies in the partial trees. As the comparison of the policy paths may terminate before it reaches their maximum lengths, the error is introduced on predicting the future policies. We bound the prediction error due to the incomplete search of the partial trees on determining approximately BE. Finally, we evaluate the empirical performance of the proposed approach in the context of multiple problem domains, and demonstrate its scalability on solving I-DIDs of significantly large horizons.

2. BACKGROUND: INTERACTIVE DID AND BEHAVIORAL EQUIVALENCE

We start with a brief review on interactive dynamic influence diagram (I-DID) and then describe its solutions that are developed using the technique on clustering behaviorally equivalent models. More details could be found in this line of research [6, 5, 19].

2.1 Interactive Dynamic Influence Diagram

I-DIDs extend probabilistic graphical models - dynamic influence diagrams (DIDs) [14] - to represent how agents make a sequence of rational decisions while interacting with other agents over time in an uncertain environment. A regular DID models sequential decision making for a single agent by linking a set of chance, decision and utility nodes over multiple time steps. To consider multiagent interaction, I-DIDs introduce a new type of node called the *model node* (hexagonal node, $M_{j,l-1}$, in Fig. 1) that represent how another agent j acts simultaneously when the modeling agent i reasons its own decisions at level l. The model node contains a set of j's candidate models at level $l-1$ ascribed by i . A link from the chance node S to the model node $M_{j,l-1}$ represents agent i's beliefs over j's models. Specifically, it is a probability distribution in the conditional probability table (CPT) of the chance node $Mod[M_j]$ (in Fig. 2). Each model, $m_{j,l-1}$, could be either a level $l-1$ I-DID or a DID at level 0. Model solutions are the predicted behavior of j and are encoded into a chance node A_j through a dashed link, called a *policy link*. Connecting A_j with other nodes in an I-DID structures how agent j's actions are engaged in i's decision making process.

Expanding an I-DID involves the update of the model node over time as indicated in the *model update link* - a dotted arrow from $M^t_{j,l-1}$ to $M^{t+1}_{j,l-1}$ in Fig. 1. As agent j acts and receives observations over time, its models are updated to reflect their changed beliefs. For each model $m^t_{j,l-1}$ at time t, its optimal solutions may include all decision options and agent j may receive any of the possible observations. Consequently, the set of updated models at time $t+1$ will have up to $|\mathcal{M}^t_{j,l-1}||A_j||\Omega_j|$ models. Here, $|\mathcal{M}^t_{j,l-1}|$ is the number of models at time step t, $|A_j|$ and $|\Omega_j|$ are the largest spaces of actions and observations respectively. The models differ in their initial beliefs updated using a configuration of action and observation. The CPT of $Mod[M^{t+1}_{j,l-1}]$ specifies the function, $\tau(b^t_{j,l-1}, a^t_j, o^{t+1}_j, b^{t+1}_{j,l-1})$ which is 1 if the belief $b^t_{j,l-1}$ in the model $m^t_{j,l-1}$ using the action a^t_j and observation o^{t+1}_j updates to $b^{t+1}_{j,l-1}$ in a model $m^{t+1}_{j,l-1}$; otherwise it is 0. We may implement the model update link using standard dependency links and chance nodes, as shown in Fig. 2, and transform an I-DID into a regular DID. Consequently, any DID technique can be exploited to solve an I-DID. Details on algorithms for solving an I-DID are in [6].

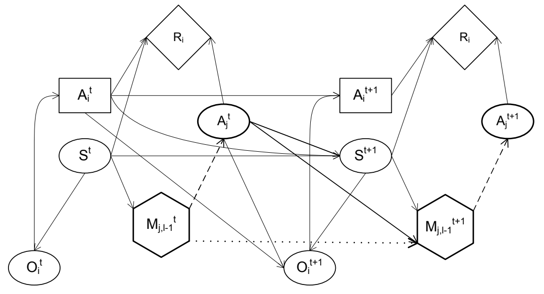


Figure 1: A generic two time-slice level l I-DID for agent i. Notice the dotted model update link that denotes the update of the models of j and of the distribution over the models, over time.

2.2 Behavioral Equivalence and Its Identification

As we may expect, the complexity of solving I-DIDs is mainly due to the growing space of possible models ascribed to other agents. It is computationally impossible if all models are considered in the model node. As the modeling agent

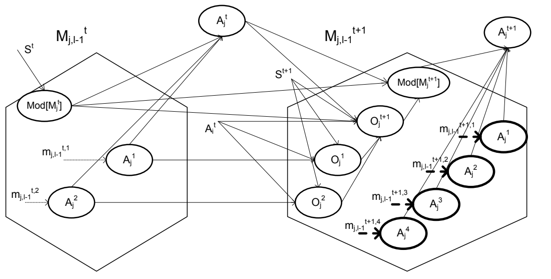


Figure 2: Implementation of the model update link using standard dependency link and chance nodes e.g. two models, $m^{t,1}_{j,l-1}$ and $m^{t,2}_{j,l-1}$, are updated into four models (shown in bold) at time $t+1$.

cares only about the predicted behavior, not the descriptive models, of the other agent, those models that have identical solutions need not be distinguished on solving I-DIDs. In other words, models that are *BE* [12] – whose behavioral predictions for the other agent are identical – could be pruned and a single representative model considered. Based on this strategy on reducing the model space, a set of algorithms have been developed with the purpose of scaling up solutions to I-DIDs over a large number of horizons [18, 5, 19]. All of the algorithms need to cope with the problem of identifying BE between a pair of models.

In the I-DID context, the other agent j's model, $m_{j,l-1}$ is a level $l-1$ I-DID or a DID if l equals to 1. Without loss of generality, we represent model solutions of T horizons as a *policy tree*, denoted by $OPT(m_{j,l-1}) \triangleq \pi^T_{m_{j,l-1}}$ where $OPT(\cdot)$ denotes the solution of the model that forms the argument. Two models, $m_{j,l-1}$ and $\hat{m}_{j,l-1}$, are BE if and only if $\pi^T_{m_{j,l-1}} = \pi^T_{\hat{m}_{j,l-1}}$. The BE identification requires to maintain and compare the entire policy trees each of which contains $(|\Omega_j|)^{T-1}$ possible paths. This is inefficient on both computational time and memory. To resolve this inefficiency, Zeng *et al.* [19] recently propose one technique to identify approximately BE by comparing depth-q ($q \leq T$) policies as well as updated beliefs, $b^{q,k}_{m_{j,l-1}}$, at the leaf nodes of the partial policy trees. Formally, let $D_{KL}[p||p']$ denote the KL divergence [11] between probability distributions, p and p'. The technique defines an approximately BE for a given measure ϵ (≥ 0).

Definition 1 ((ϵ, q)-BE). *Two models of agent j, $m_{j,l-1}$ and $\hat{m}_{j,l-1}$, are (ϵ,q)-BE, $\epsilon \geq 0$, $q \leq T$, if their depth-q policy trees are identical, $\pi^q_{m_{j,l-1}} = \pi^q_{\hat{m}_{j,l-1}}$, and if $q < T$ then beliefs at the leaves of the two policy trees diverge by at most ϵ: $\max\limits_{k=1\ldots|\Omega_j|^q} D_{KL}[b^{q,k}_{m_{j,l-1}}||b^{q,k}_{\hat{m}_{j,l-1}}] \leq \epsilon$.*

More importantly, it is found that the depth q of the partial tree can be determined given some ϵ. Eq. 1 shows the way of computing q, where γ_F is a minimal mixing rate in a stochastic transition and $b^{0,k}_{m_{j,l-1}}$ ($b^{0,k}_{\hat{m}_{j,l-1}}$)) initial beliefs in j's model $m_{j,l-1}$ ($\hat{m}_{j,l-1}$). The computation is based on the fact: the KL divergence between the distributions over the same space contacts with the rate $(1-\gamma_F)$ after one transition [1]. In the I-DID context, γ_F is the minimum probability mass on some state due to the transition, and is computed by multiplying the state transition probability and the likelihood of observation for j.

$$q = min\left\{T, max\{0, \lfloor\frac{ln\frac{\epsilon}{D_{KL}(b^{0,k}_{m_{j,l-1}}||b^{0,k}_{\hat{m}_{j,l-1}})}}{ln(1-\gamma_F)}\rfloor\}\right\} \quad (1)$$

Accordingly, the straightforward implementation for identifying (ϵ, q)-BE is to firstly build partial policy trees of depth-q (line 2) and then check the equality between them (line 3). It is called as a plain algorithm for identifying (ϵ, q)-BE of two models, ϵ-BE-P, as shown in Fig. 3.

```
ε-BE-P (Models, $m_{j,l-1}$ and $\hat{m}_{j,l-1}$, Horizon $T$, and
parameters, $\gamma_F$ and $\epsilon$)

  1. Compute $q$ according to Eq. 1
  2. Build depth-$q$ partial trees: $\pi^q_{m_{j,l-1}}$ and $\pi^q_{\hat{m}_{j,l-1}}$
  3. If $\pi^q_{m_{j,l-1}} = \pi^q_{\hat{m}_{j,l-1}}$
  4.   Return $True$; Else, Return $False$
```

Figure 3: A plain algorithm, ϵ-BE-P, for approximate BE identification by comparing the entire depth-q trees.

3. INCREMENTAL BE IDENTIFICATION

As discussed above, identifying the behaviorally equivalent models of the other agent j plays a central role in the I-DID solutions. Using partial policy trees provides a promising direction to scale BE to large horizons since it groups together more models that could be approximately BE and simplifies the complexity of identifying BE by comparing only a subset of the entire policy trees. A plain realization of this strategy is to compare the partial policy trees that are symmetric and are fully constructed using a uniform length for all policy paths. We aim to further enhance the use of partial policies to cluster more approximately BE models in a more efficient way. We firstly define approximately BE models using asymmetric policy trees and then propose an incremental technique to identify the models.

3.1 Approximate BE

A q-length policy path is an action-observation sequence describing what agent j acts and observes over q time steps. It is denoted by, $h^q_j = \{a^t_j, o^{t+1}_j\}^q_{t=1}$, where o^{T+1}_j is null for a T ($q \leq T-1$) horizon planning problem. If $a^t_j \in A_j$ and $o^{t+1}_j \in \Omega_j$, where A_j and Ω_j are agent j's action and observation sets respectively, then a depth-q policy tree is a set of all q-length paths: $\pi^q_j = \Pi^q_1(A_j \times \Omega_j)$ where o^q_j is null. As we may notice, the tree is symmetric since all paths have the same length q. For an asymmetric policy tree, we need to enumerate the set of policy paths and some paths may differ in the length. We index paths of the same length by imposing an order on the observations in the policy tree. Formally, let $\pi^{q_L,q_U}_j =< h^{q_1,1}_j, \cdots, h^{q_r,k}_j >$ be the asymmetric policy tree of depth-(q_L, q_U) where q_L is the minimum length, $q_L = Min(q_1, \cdots, q_r)$, q_U the maximum one, $q_U = Max(q_1, \cdots, q_r)$, and k an index number.

Notice that beliefs updated using an action-observation sequence in a partially observable stochastic process is a sufficient statistic for the history. Consequently, future policies are predicted only on the updated beliefs. If $b^{0,k}_{j,l-1}$ is the initial belief in the model, $m_{j,l-1}$, then let $b_{j,l-1}^{q,k}$ be the new belief on updating it using the q-length policy path $h_j^{q,k}$. The policies, $\Pi_{q+1}^T(A_j \times \Omega_j)$, succeeding to the path $h_j^{q,k}$ can be predicted using the belief $b_{j,l-1}^{q,k}$. Using the partial trees and updated beliefs, we may re-write the full policy tree as follows: $\pi_{m_{j,l-1}}^T = < \pi_{m_{j,l-1}}^{q_L,q_U}, B_{m_{j,l-1}}^{q_L,q_U} >=< (h_j^{q_1,1}, b_{m_{j,l-1}}^{q_1,1}), \cdots, (h_j^{q_r,k}, b_{m_{j,l-1}}^{q_r,k}) >$, where $B_{m_{j,l-1}}^{q_L,q_U}$ is the set of updated beliefs. Consequently, comparing a small number of policy paths and beliefs is sufficient to identify BE. We modify Def. 1 to formulate an approximately BE, called (ϵ, q_L, q_U)-BE, between models as follows.

DEFINITION 2 ((ϵ, q_L, q_U)-BE). *Two models of agent j, $m_{j,l-1}$ and $\hat{m}_{j,l-1}$, are (ϵ, q_L, q_U)-BE, $\epsilon \geq 0$, $q_U \leq T$, if their depth-(q_L, q_U) policy trees are identical, $\pi_{m_{j,l-1}}^{q_L,q_U} = \pi_{\hat{m}_{j,l-1}}^{q_L,q_U}$, and if $q_U < T$ then updated beliefs for the two policy trees diverge by at most ϵ:*

$$\max_{(q_1,1),\cdots,(q_r,k)} D_{KL}[b_{m_{j,l-1}}^{q_r,k} || b_{\hat{m}_{j,l-1}}^{q_r,k}] \leq \epsilon.$$

Intuitively, two models are (ϵ, q_L, q_U)-BE if they have identical solutions of depth-(q_L, q_U) trees and the divergence of pairs of the updated beliefs at the leaves of the depth-(q_L, q_U) tree is not larger than ϵ. Two (ϵ, q_L, q_U)-BE models become exact BE as ϵ approaches zero. If the partial tree is symmetric in the setting of $q_L = q_U$, (ϵ, q_L, q_U)-BE is equivalent to the notion of approximately BE in Def. 1. Hence (ϵ, q_L, q_U)-BE provides a general definition of approximately BE using asymmetric partial trees. The remaining question is how to compute values for the parameters, q_L and q_U, given some ϵ.

As mentioned in Sec. 2.2, the mixing rate is computed as the minimal one for the transitions of all possible action-observation pairs. Eq. 1 provides a principled way of determining the maximum length q_U for all policy paths given the amount of approximation ϵ. Meanwhile, we observe that the divergence of updated beliefs using some paths may turn out to be much less than ϵ before the paths are fully extended into the length q_U. This may occur due to the fact that the KL divergence of belief distributions contracts monotonically over time [1]. The minimum length q_L is the earliest time when the belief divergence is known to be smaller than ϵ. Its value is found during the BE identification.

3.2 Incremental Comparison

(ϵ, q_L, q_U)-BE provides a novel way to identify approximately BE and compares partial trees with an asymmetric structure. This differs from (ϵ, q)-BE that needs to compare a full size of partial trees. Due to the unknown value for the minimum length, the size of asymmetric trees can't be decided given a single input of approximation measure ϵ. However, we are able to bound the tree size using the maximum path length, which avoids an arbitrary expansion on the tree. For the purpose of identifying (ϵ, q_L, q_U)-BE, we propose an incremental technique below.

We compare both partial trees and updated beliefs at the leaves of the trees when we expand the policy tree at every time step. We terminate the comparison once there is any unmatched behavior in the paths; otherwise, we expand the trees until the depth of the partial trees reaches the maximum value q_U. In addition, we do not further expand a partial tree at the end of a policy path if the path is identical and the divergence of updated beliefs is not larger than ϵ. This is because the equivalence of future behavior can be sufficiently determined without checking the unexpanded partial trees. The q_L value is the minimum length of all paths when the comparison terminates. The procedure is an incremental policy comparison for the (ϵ, q_L, q_U)-BE identification, called ϵ-BE-I. We illustrate the procedure using an example in Fig. 4. The example is constructed in the *Tiger* problem domain - well studied in the POMDP literature.

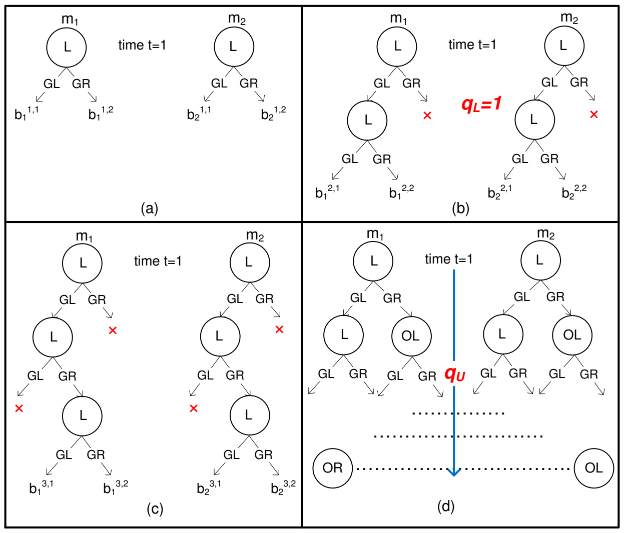


Figure 4: ϵ-BE-I, (a)-(c), and ϵ-BE-P, (d), for (ϵ, q_L, q_U)-BE identification.

Example: We are checking whether two models of agent j, m_1 and m_2, are (ϵ, q_L, q_U)-BE given the approximation amount ϵ. Assuming that b_1^0 and b_2^0 are their initial beliefs and the mixing rate γ_F is computed in the domain, we calculate the q_U value [1] in Eq. 1 that will serve as the upper bound for iterating the comparison. Since $D_{KL}(b_1^0||b_2^0)$ is larger than ϵ, we need to build the root nodes for the policy trees that are solutions of two models respectively. We then update their initial beliefs into new ones given possible observations (GL and GR) because both trees have identical actions L at time t=1 (Fig. 4(a)). We compute the divergences of each pair of new beliefs given the same observation like $D_{KL}(b_1^{1,1}||b_2^{1,1})$ and $D_{KL}(b_1^{1,2}||b_2^{1,2})$. Suppose that $D_{KL}(b_1^{1,1}||b_2^{1,1})$ is still larger than ϵ while $D_{KL}(b_1^{1,2}||b_2^{1,2})$ is less than ϵ. We must expand the policy tree following the path $\{L, GL\}$, but will not continue the expansion in the other path $\{L, GR\}$ at t=2 (Fig. 4(b)). We say that this path is *blocked* (denoted by $\times$) and will not be considered for a further expansion. The minimum path length q_L is now found and is equal to 1. We compare the policies following the path $\{L, GL\}$, and update their beliefs if the policies are equivalent at t=2. We repeat the same procedure at t=3 and so on until either the maximal depth q_U is approached or no policy paths can be further expanded (Fig. 4(c)). The incremental procedure may generate a small size of the partial policy trees for identifying the equivalence between m_1 and m_2. For the same identification purpose, the previous algorithm, ϵ-BE-P, needs to construct and compare the partial policy trees that expand all paths to the maximal length

[1]We may predefine the q_U value if it can't be computed in Eq. 1

q_U (Fig. 4(d)).

In addition, we observe that the depth value q can't be calculated in Eq. 1 if the mixing rate γ_F becomes zero. To run the ϵ-BE-P algorithm, we need to specify the q value even given the known approximation amount ϵ. This results in a partial policy tree that is arbitrarily large. We need to check the equality for all paths in the partial tree for the identification purpose. On the other hand, the incremental algorithm, ϵ-BE-I, employs ϵ as the threshold value to prune the partial trees while it performs the path comparison and expands the trees. The predefined depth q_U acts as an upper bound value to terminate the identification process if it is necessary. In summary, the incremental policy comparison algorithm becomes a universal approach for identifying approximately BE models.

3.3 Algorithm

We present the incremental policy comparison algorithm for identifying (ϵ, q_L, q_U)-BE between two models in Fig. 5. As mentioned in the previous section, the algorithm terminates the identification process when any of the following conditions is met: (a) Initial beliefs diverge at most ϵ and (ϵ, q_L, q_U)-BE of two models are immediately ascertained (line 6); (b) Any unmatched policy is detected and the models are not (ϵ, q_L, q_U)-BE (line 11); (c) (ϵ, q_L, q_U)-BE is confirmed for two models when either no path can be further expanded or the depth q_U is approached (lines 12-14). By doing this, we can avoid the expansion of entire depth-q_U trees while achieving the identification of (ϵ, q_L, q_U)-BE between two models.

```
ε-BE-I (Models, $m_{j,l-1}$ and $\hat{m}_{j,l-1}$, Horizon $T$, and
parameters, $\gamma_F$ and $\epsilon$)

 1. Case $\gamma_F \in (0,1]$: Compute $q_U$ according to Eq. 1
 2. Case $\gamma_F$=0: Specify $q_U \leq T$
 3. For $t$=1 to $q_U$ do
 4.     If $D_{KL}(b^{t-1,k}_{m_{j,l-1}} || b^{t-1,k}_{\hat{m}_{j,l-1}}) \leq \epsilon$
 5.         Case $t > 1$: Block the path $h^{t,k}_{m_{j,l-1}}$ ($h^{t,k}_{\hat{m}_{j,l-1}}$)
 6.         Case $t$=1: Return $True$ and Break
 7.     else
 8.         If $h^{t,k}_{m_{j,l-1}} = h^{t,k}_{\hat{m}_{j,l-1}}$
 9.             Case $t<T$: Expand the $t$-length paths and
                compute the updated belief $b^{t,k}_{m_{j,l-1}}$ ($b^{t,k}_{\hat{m}_{j,l-1}}$) given
                the path $h^{t,k}_{m_{j,l-1}}$ ($h^{t,k}_{\hat{m}_{j,l-1}}$)
10.             Case $t$=$T$: Return $True$
11.         else Return $False$ and Break
12.     If All paths $h^{t,k}_{m_{j,l-1}}$ ($h^{t,k}_{\hat{m}_{j,l-1}}$) are blocked
13.         Return $True$
14. Return $True$
```

Figure 5: An incremental algorithm, ϵ-BE-I, for determining the equivalence of two models given the approximate amount ϵ.

ϵ-BE-I differs from ϵ-BE-P since it compares only a subset of depth-q_U trees. It blocks the path $h^{q_r,k}_{m_{j,l-1}}$ ($h^{q_r,k}_{\hat{m}_{j,l-1}}$) for a further comparison when the divergence of beliefs, $D_{KL}(b^{q_r,k}_{m_{j,l-1}} || b^{q_r,k}_{\hat{m}_{j,l-1}})$, is smaller than ϵ at time step q_r. Notice that the partial trees succeeding to $h^{q_r,k}_{m_{j,l-1}}$ ($h^{q_r,k}_{\hat{m}_{j,l-1}}$) may not be identical although the updated beliefs have a small amount of divergence. Consequently, ϵ-BE-I may result in grouping more approximately BE than ϵ-BE-P.

4. COMPUTATIONAL SAVINGS AND ERROR BOUND

Algorithms for determining (ϵ, q_L, q_U)-BE of a pair of models mainly perform the path comparison in policy trees. The complexity is proportional to the number of comparisons required to approximately decide the equivalence. For the ϵ-BE-P algorithm, we need to compare every path in partial trees of depth-q. Since there are a maximum of $|\Omega_j|^{q_U-1}$ leaf nodes in a depth-q_U tree, the complexity of ϵ-BE-P is $\mathcal{O}(|\mathcal{M}_{j,l-1}|^2|\Omega_j|^{q_U})$ where $|\mathcal{M}_{j,l-1}|$ is the number of candidate models. On the other hand, the ϵ-BE-I algorithm prunes the paths while it traverses a depth-q_U tree from the root. This may result in an asymmetric partial tree where the number of leaf nodes is N where $N \ll |\Omega_j|^{q_U-1}$. Meanwhile the ϵ-BE-I algorithm needs to compare beliefs for which the number is also bounded by N. Consequently, the complexity of ϵ-BE-I becomes $\mathcal{O}(2|\mathcal{M}_{j,l-1}|^2 N)$. In addition, ϵ-BE-I involves the belief calculation in the procedure that costs little on propagating beliefs in solved models.

Both algorithms preclude storing entire policy trees that contain $(|\Omega_j|)^{T-1}$ possible paths. For the ϵ-BE-P algorithm, we maintain at most $2(|\Omega_j|)^{q_U-1}$ paths ($q_U \leq T$) at each time step when a pair of models are under the identification. For the ϵ-BE-I algorithm, we need to store only $2N$ paths each of which has the length bounded by q_U. Hence ϵ-BE-I achieves much better memory efficiency compared to ϵ-BE-P.

We analyze the error in the value of j's predicted behavior. An error occurs when a behaviorally distinct model, $m_{j,l-1}$, is grouped with the model, $\hat{m}_{j,l-1}$, given an approximation amount ϵ. Let $m_{j,l-1}$ be the model associated with $\hat{m}_{j,l-1}$, resulting in the worst error. Let α_T and $\hat{\alpha}_T$ be the exact entire policy trees obtained by solving the two models, respectively. Then, the error is: $\rho = |\alpha^T \cdot b^0_{m_{j,l-1}} - \alpha^T \cdot b^0_{\hat{m}_{j,l-1}})|$. As ϵ-BE-I starts to prune the path at the length q_L. The error in the worst case becomes:

$$\begin{array}{rcl}
\rho & = & |\alpha^{T-q_L} \cdot b^{q_L}_{m_{j,l-1}} - \alpha^{T-q_L} \cdot b^{q_L}_{\hat{m}_{j,l-1}}| \\
& = & |\alpha^{T-q_L} \cdot b^{q_L}_{m_{j,l-1}} + \hat{\alpha}^{T-q_L} \cdot b^{q_L}_{m_{j,l-1}} - \hat{\alpha}^{T-q_L} \cdot b^{q_L}_{m_{j,l-1}} \\
& & -\alpha^{T-q_L} \cdot b^{q_L}_{\hat{m}_{j,l-1}}| \quad \text{(add zero)} \\
& \leq & |\alpha^{T-q_L} \cdot b^{q_L}_{m_{j,l-1}} + \hat{\alpha}^{T-q_L} \cdot b^{q_L}_{\hat{m}_{j,l-1}} - \hat{\alpha}^{T-q_L} \cdot b^{q_L}_{m_{j,l-1}} \\
& & -\alpha^{T-q_L} \cdot b^{q_L}_{\hat{m}_{j,l-1}}| \quad (\hat{\alpha}^{T-q_L} \cdot b^{q_L}_{\hat{m}_{j,l-1}} \geq \hat{\alpha}^{T-q_L} \cdot b^{q_L}_{m_{j,l-1}}) \\
& = & |(\alpha^{T-q_L} - \hat{\alpha}^{T-q_L}) \cdot (b^{q_L}_{m_{j,l-1}} - b^{q_L}_{\hat{m}_{j,l-1}})| \\
& & \text{(Hölder's ineq.)} \\
& \leq & |\alpha^{T-q_L} - \hat{\alpha}^{T-q_L}|_\infty \cdot |(b^{q_L}_{m_{j,l-1}} - b^{q_L}_{\hat{m}_{j,l-1}})|_1 \\
& & \text{(Pinsker's ineq.)} \\
& \leq & |\alpha^{T-q_L} - \hat{\alpha}^{T-q_L}|_\infty \cdot 2D_{KL}(b^{q_L}_{m_{j,l-1}} || b^{q_L}_{\hat{m}_{j,l-1}}) \\
& \leq & (R_j^{max} - R_j^{min})(T - q_L) \cdot 2\epsilon \quad \text{(by definition)}
\end{array}$$

Here, R_j^{max} and R_j^{min} are the maximum and minimum rewards of j, respectively. This error bound is not tight as that of ϵ-BE-P which is $(R_j^{max} - R_j^{min})(T - q_U) \cdot 2\epsilon$ when $q_L < q_U$. The gap is due to the utilization of ϵ on approximating BE at different depths as previously mentioned. We expect that the subtle difference has a limited impact on the I-DID solutions by clustering approximately BE models.

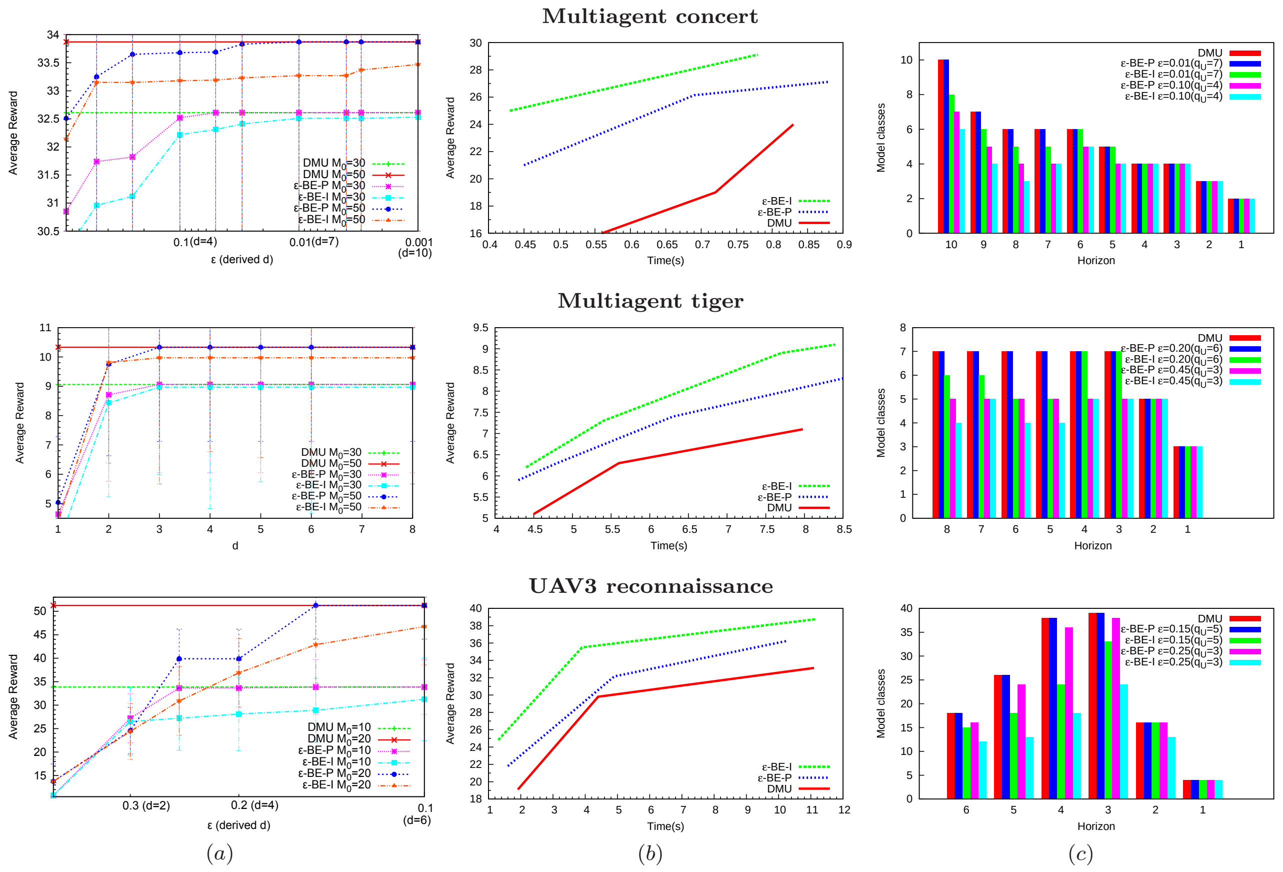


Figure 6: Performance profile obtained by solving level 1 I-DIDs for the different problem domains using ϵ-BE-I, ϵ-BE-P and DMU. (a) Average rewards; (b) Efficiency comparison; and (c) Reduced model space.

5. EXPERIMENTAL RESULTS

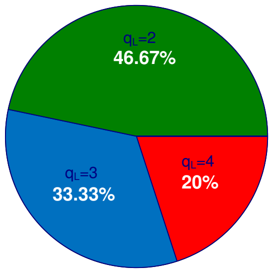
We implemented ϵ-BE-I algorithm for determining (ϵ, q_L, q_U)-BE of models and use it to group models into a class. We then select one representative model for each class while pruning others, similarly to using exact BE. We embed the procedure into the algorithm for solving I-DIDs. We also compare it with ϵ-BE-P algorithm (letting q=q_U=q_L) which serves as a baseline on approximating (ϵ, q_L, q_U)-BE. In addition, we compare both algorithms with one exact BE approach, called discriminative model update (DMU), previously proposed to solve I-DIDs [5]. DMU approach clusters BE models by comparing their entire policy trees and updates only those models that will be behaviorally distinct from existing ones. We evaluate all of these three approaches (namely ϵ-BE-I, ϵ-BE-P and DMU) when they are used to solve level 1 I-DIDs of increasing horizons over four problem domains. Relevant information on domain dimensions and minimal mixing rates are listed in Table 1. Note that UAV5 - an extended version of the two-agent unmanned aerial vehicle (UAV) problem [4, 19] - is the largest domain so far used to evaluate the I-DIDs.

We formulate level 1 I-DIDs of increasing horizons for the problems and solve them using the three approaches. We show that the quality of the policies generated by ϵ-BE-I approaches that of ϵ-BE-P given the same approximation measure. Meanwhile, the solution quality generated by both approximate techniques converges to that of the exact DMU as ϵ decreases (with the corresponding increase in q_U). We also show that in most cases ϵ-BE-I is able to identify approximately BE models without constructing the partial trees of a full size. This verifies the utility of using the incremental technique for the BE identification purpose. In addition, we demonstrate that ϵ-BE-I further reduces the model space and performs better than ϵ-BE-P on the issue of solution scalability.

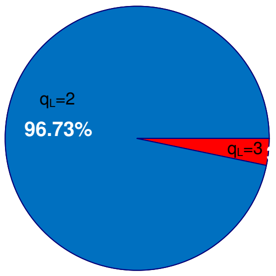
Domains	γ_F	$\|S\|$	$\|A_i\|$	$\|A_j\|$	$\|\Omega_i\|$	$\|\Omega_j\|$
Tiger [9]	0	2	3	3	6	3
UAV3 [4, 19]	0.2	25	5	5	4	5
Concert [19]	0.5	2	3	3	4	2
UAV5	0.2	81	5	5	4	5

Table 1: Domains used to evaluate algorithms for solving I-DIDs.

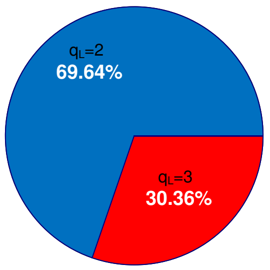
In Fig. 6(a), we report the average rewards gathered by simulating the I-DID solutions over 1,000 runs. Each run of simulation is executed by randomly picking up the true model of j according to i's belief. We used a horizon of 10 for the *Concert* domain, 8 for the *Tiger* and 6 for the *UAV3*. For a given number of initial models $M_{j,0}$, ϵ-BE-I obtains similar average rewards in comparison to ϵ-BE-P. As expected, their solutions improve and converge toward the exact method DMU as ϵ reduces.



(a) **Multiagent concert (T=10)**



(b) **Multiagent tiger (T=8)**



(c) **UAV3 reconnaissance (T=6)**

Figure 7: Likelihood that ϵ-BE-I terminates at q_L in the setting of (a) q_U=5; (b) q_U=6; and (c) q_U=4.

Fig. 6(b) confirms our intuition on the favorable efficiency of ϵ-BE-I technique. For a given allocated time, ϵ-BE-I obtains larger rewards than other approaches including both DMU and ϵ-BE-P. As we show in Fig. 6(c), the number of models in a model node drops when ϵ-BE-I is employed to prune the model space. This is because ϵ-BE-I clusters more approximately BE models using a small set of policies.

In Fig. 7, we show the likelihood that ϵ-BE-I terminates the comparison before reaching the maximum length, q_U, of all paths in the partial policy trees. We compute this likelihood as the percentage of occurrences for each q_L value under a given ϵ. The cases, $q_L < q_U$, are often observed to terminate the policy comparison especially for a small ϵ value (with the corresponding large q_U value).

In Table 2, we show the running times of three techniques for solving problems of increasing horizons. In obtaining the run times for the approximations, we adjusted the corresponding parameters so that the quality of the solution by each approach was similar to each other. ϵ-BE-I achieves the reduced running times and improved scalability over all domains. In particular, for the large UAV5 domain, we were able to solve the I-DIDs for more than 8 time steps.

Level 1	**T**	**Time (s)**		
		DMU	ϵ-BE-I	ϵ-BE-P
Concert	6	0.29	0.11	0.31
	10	2.3	0.22	1.9
	25	*	**9.1**	**13.1**
Tiger	6	0.34	0.16	0.21
	8	1.3	0.21	0.37
	20	*	**2.49**	**3.1**
UAV3	6	13.1	8.1	8.9
	8	161	19	27
	10	*	48	55
	20	*	76	98
	25	*	**132**	*
UAV5	4	19.3	7.9	9.8
	6	*	16	31
	8	*	**60**	*

Table 2: ϵ-BE-I scales better than other approaches. Experiments were run on a Linux platform with Intel Core2 2.4GHz with 4GB of memory.

6. RELATED WORK

I-DIDs [6] emerge as an important framework on modeling multiagent decision making problems. Models for the similar purpose include multiagent influence diagrams (MAIDs) [10], and networks of influence diagrams (NIDs) [7, 8]. These formalisms structure the complex problem domains by decomposing the situation into chance and decision variables, and the dependencies between the variables. MAIDs objectively analyze the game, efficiently computing the Nash equilibrium profile by exploiting the independence structure. NIDs extend MAIDs to include agents' uncertainty over the game being played and over models of the other agents. Both MAIDs and NIDs provide an analysis of the game from an external viewpoint, and adopt Nash equilibrium as the solution concept. However, equilibrium is not unique – there could be many joint solutions in equilibrium with no clear way to choose between them – and incomplete – the solution does not prescribe a policy when the policy followed by the other agent is not part of the equilibrium. Specifically, MAIDs do not allow us to define a distribution over non-equilibrium behaviors of other agents. Furthermore, their applicability is limited to static single play games. Interactions are more complex when they are extended over time, where predictions about others' future actions must be made using models that change as the agents act and observe. I-DIDs seek to address this gap by offering an intuitive way to extend sequential decision making as formalized by DIDs to multiagent settings.

As we mentioned before, the complexity of I-DIDs is mainly due to the exponential growth in the candidate models over time. Using the insight that models whose beliefs are spatially close are likely to be behaviorally equivalent, Zeng *et al.* [18] employed a k-means approach to cluster models together and select K representative models in the model node at each time step. This approach needs to expand all models before clustering is applied, which consumes a large amount of memory on storing the models. A recent approach [5] preemptively avoids expanding models that will turn out to be behaviorally equivalent to others in the new time step. By discriminating between model updates, the approach generates a minimal set of models in each non-initial model node. This line of work exploits the concept of BE, introduced earlier [13, 12]. The developed method quickly turns to be inefficient since it requires to maintain and compare the entire policy trees for identifying BE models. In parallel, Zeng *et al.* [15] attempted to cluster models using K most probable paths in the policy tree. However, the proposed technique is facing an unsolved problem on computing path probabilities. Similarly, the attempt using subjectivly equivalent models to cluster models requires the prediction on behavior of the modeling agent [3]. Another efficient way to reduce the model space is achieved by clustering models that are actionally equivalent [16]. Recently, Zeng and Doshi [17] compare various I-DID solutions and demonstrate their utilities in more problem domains.

7. DISCUSSION

I-DIDs provide a graphical formalism for modeling the sequential decision making of an agent in an uncertain multiagent setting. The increased complexity of I-DIDs is predominantly due to the large space of candidate models and its exponential growth over time. Previous solutions to I-DIDs limit the model growth mainly by clustering BE models at each step. We presented an improved version of using partial policies to identify BE models. We defined approximately BE based on a partial policy tree that has an asymmetric structure and allows different lengths for its paths. Our definition avoids building a full size of the partial trees and clusters more models that are approximately BE. We showed that our new approach gains much computational savings and achieves better scalability over the state of the art approach. As we note that our approach is developed based on the contraction property of problem domains, we may further refine the approach by exploiting the relevant property.

8. ACKNOWLEDGMENT

The authors acknowledge the supports from NSFC (#60974089 and #60975052). Yifeng thanks Dr. Prashant Doshi (in the University of Georgia) for useful comments on the first draft.

9. REFERENCES

[1] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *The 14th Conference on Uncertainty in Artificial Intelligence(UAI)*, pages 33–42, 1998.

[2] E. Dekel, D. Fudenberg, and S. Morris. Topologies on types. *Theoretical Economics*, 1:275–309, 2006.

[3] P. Doshi, M. Chandrasekaran, and Y. Zeng. Epsilon-subject equivalence of models for interactive dynamic influence diagrams. In *WIC/ACM/IEEE Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 165–172, 2010.

[4] P. Doshi and E. Sonu. Gatac: A scalable and realistic testbed for multiagent decision making. In *AAMAS 2010 Workshop on Multi-agent Sequential Decision-Making in Uncertain Domains*, pages 64–68, 2010.

[5] P. Doshi and Y. Zeng. Improved approximation of interactive dynamic influence diagrams using discriminative model updates. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 907–914, 2009.

[6] P. Doshi, Y. Zeng, and Q. Chen. Graphical models for interactive pomdps: Representations and solutions. *Journal of Autonomous Agents and Multiagent Systems (JAAMAS)*, 18(3):376–416, 2009.

[7] K. Gal and A. Pfeffer. Networks of influence diagrams: A formalism for representing agents' beliefs and decision-making processes. *Journal of Artificial Intelligence Research*, 33:109–147, 2008.

[8] Y. Gal and A. Pfeffer. A language for modeling agent's decision-making processes in games. In *Autonomous Agents and Multi-Agents Systems Conference (AAMAS)*, pages 265–272, 2003.

[9] P. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multiagent settings. *Journal of Artificial Intelligence Research (JAIR)*, 24:49–79, 2005.

[10] D. Koller and B. Milch. Multi-agent influence diagrams for representing and solving games. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1027–1034, 2001.

[11] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 1951.

[12] D. Pynadath and S. Marsella. Minimal mental models. In *Twenty-Second Conference on Artificial Intelligence (AAAI)*, pages 1038–1044, Vancouver, Canada, 2007.

[13] B. Rathnas., P. Doshi, and P. J. Gmytrasiewicz. Exact solutions to interactive pomdps using behavioral equivalence. In *Autonomous Agents and Multi-Agents Systems Conference (AAMAS)*, pages 1025–1032, 2006.

[14] J. A. Tatman and R. D. Shachter. Dynamic programming and influence diagrams. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2):365–379, 1990.

[15] Y. Zeng, Y. Chen, and P. Doshi. Approximating behavioral equivalence of models using top-k policy paths. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1229–1230, 2011.

[16] Y. Zeng and P. Doshi. Speeding up exact solutions of interactive influence diagrams using action equivalence. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1996–2001, 2009.

[17] Y. Zeng and P. Doshi. Exploiting model equivalences for solving interactive dynamic influence diagrams. *Journal of Artificial Intelligence Research (JAIR)*, 43:211–255, 2012.

[18] Y. Zeng, P. Doshi, and Q. Chen. Approximate solutions of interactive dynamic influence diagrams using model clustering. In *Twenty Second Conference on Artificial Intelligence (AAAI)*, pages 782–787, Vancouver, Canada, 2007.

[19] Y. Zeng, P. Doshi, Y. Pan, H. Mao, M. Chandrasekaran, and J. Luo. Utilizing partial policies for identifying equivalence of behavioral models. In *The Twenty-Fifth Conference on Artificial Intelligencee(AAAI)*, pages 1083–1088, 2011.

Learning and Reasoning about Norms using Neural-Symbolic Systems

Guido Boella[1], Silvano Colombo Tosatto[1,2], Artur D'Avila Garcez[3],
Valerio Genovese[1,2], Alan Perotti[1], and Leendert van der Torre[2]

[1]University of Turin, Italy. {*guido, genovese, perotti*}*@di.unito.it*

[2]CSC, University of Luxembourg. {*silvano.colombotosatto, leon.vandertorre*}*@uni.lu*

[3]City University London. *aag@soi.city.ac.uk*

ABSTRACT

In this paper we provide a neural-symbolic framework to model, reason about and learn norms in multi-agent systems. To this purpose, we define a fragment of Input/Output (I/O) logic that can be embedded into a neural network. We extend d'Avila Garcez et al. Connectionist Inductive Learning and Logic Programming System (CILP) to translate an I/O logic theory into a Neural Network (NN) that can be trained further with examples: we call this new system Normative-CILP (N-CILP). We then present a new algorithm to handle priorities between rules in order to cope with normative issues like Contrary to Duty (CTD), Priorities, Exceptions and Permissions. We illustrate the applicability of the framework on a case study based on RoboCup rules: within this working example, we compare the learning capacity of a network built with N-CILP with a non symbolic neural network, we explore how the initial knowledge impacts on the overall performance, and we test the NN capacity of learning norms, generalizing new Contrary to Duty rules from examples.

Categories and Subject Descriptors

H.4.m [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms, Experimentation, Theory, Legal Aspects

Keywords

Knowledge representation, Single agent reasoning, Computational architectures for learning, Single agent learning

1. INTRODUCTION

In artificial social systems, norms are mechanisms to effectively deal with coordination in normative multi-agent systems (MAS). An open problem in AI is how to equip agents to deal effectively with norms that change over time [3], either due to explicit changes made by legislators or due to different interpretations of the law by judges and referees.

In this paper we combine Input/Output (I/O) logic [11] with the neural-symbolic paradigm [7] in order to address the following research question:

- *How to define a formal framework for reasoning and learning about norms in a dynamic environment?*

Input/Output (I/O) logic [11] is a symbolic formalism used to represent and reason about norms. I/O logic provides some reasoning mechanisms to produce outputs from the inputs, and each of them bears a specific set of features.

The neural-symbolic paradigm of [7] embeds symbolic logic into neural networks. Neural-symbolic systems provide translation algorithms from symbolic logic to neural networks and vice-versa. The resulting network is used for robust learning and computation, while the logic provides (i) background knowledge to help learning (as the logic is translated into the NN) and (ii) high-level explanations for the network models (when the trained NN is translated into the logic).CILP is an advanced neural-symbolic systems and it has been shown an effective tool in exploiting symbolic background knowledge (i.e. on incomplete domain theory) with learning from examples.

We study how to represent I/O within the computational model of neural networks (NNs). We choose I/O logic because it presents a strong similarity with NNs: both have a separate specification of inputs and outputs. We exploit this analogy to encode symbolic knowledge (expressed in terms of I/O rules) into NNs, and then we use the NN to reason and learn new norms in a dynamic environment.
Hence two Research sub-Questions are:

- *How to represent I/O logic rules in neural networks?*
- *How to refine normative rules and learn new ones?*

Below, we define the language used to express norms and we present an extension of the "Connectionist Inductive Learning and Logic Programming" system (CILP) [7], called Normative-CILP (N-CILP).

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

With the exception of game-theoretic approaches [17, 5, 18], few machine learning techniques have been applied to tackle open problems like learning and/or revising new norms in open and dynamic environments.

We show how to use NNs to cope with some of the underpinnings of normative reasoning: *permissions*, *contrary to duties* (CTD) and *exceptions* by using the concept of priorities between the rules.

We also tested our tool on a case study based on the RoboCup competition, representing a significant set of the *rules of the game* from [13] in I/O logic and then studying the capability of the tool in learning new norms and performing reasoning. The results show that the I/O encoding improves the capacity of the NN of learning norms.

The contribution of this work is in studying and combining symbolic and sub-symbolic representations to provide a flexible and effective methodology for learning, normative reasoning and specification in MAS. In this process, we have also made a contribution to the area of neural-symbolic integration: by studying neural-symbolic systems from the point of view of normative reasoning we have been able to propose a new translation of priorities into object-level negation. From a theoretical perspective, we are interested in studying the similarities between I/O logic and neural networks. From a practical point of view, it is hoped that the network model will lead directly to an efficient hardware implementation. The normative CILP tool has been implemented in Java and is available for download (together with the dataset) at *http://www.di.unito.it/∼genovese/tools/NNSS.zip*. The experiments reported here indicate how promising is this line of research.

The paper is structured as follows: In section 2 we describe the relevant background about the neural-symbolic approach, I/O logic and normative agents. In Section 3 we introduce our approach and a motivating example. In Section 4 we show how to encode I/O logic into a neural-network using the Normative-CILP translation algorithm. In Section 5 we present and discuss the results obtained from the experiments. Section 6 concludes the paper and discusses directions for future work.

2. RELATED WORK

2.1 Neural-Symbolic approach

The main purpose of a *neural-symbolic approach* is to bring together connectionist and symbolic approaches [7]. In this way it is possible to exploit the strengths of both approaches and hopefully avoid their drawbacks. With such approach we are able to formally represent the norms governing the normative system in a neural network. In addition we are also capable of exploiting the instance learning capacities of neural networks and their massive parallel computation.

Algorithms like *KBANN*[19] and *CILP*[8] provide a translation of a symbolic representation of knowledge into a neural network. The advantage of CILP is that it uses a provably sound translation into single-hidden layer networks with sigmoid activation functions. This allows the efficient use of *backpropagation* for learning. In what follows, we use a variant of CILP since we are interested in the integration of reasoning and learning capabilities.

2.2 I/O Logic

To describe the norms regulating the system we use I/O Logic [11]. Rules used in I/O logic are defined as couples $R_1 = (A, B)$, where both A and B represent sets of literals that can be in disjunctive or conjunctive form. A is called the *antecedent* of the rule, while B is the *consequent*: A must hold for the rule to be activated, and B is consequently activated. I/O logic provides some reasoning mechanisms to produce outputs from the inputs, and each of them bears a specific set of features. The *simple-minded output* does not satisfy the principle of identity, but it allows the *strengthening input*, *conjoining output* and *weakening output* features. The *basic output* and *reusable output* mechanisms allow the additional features of *input disjunction* and *reusability*, while the *reusable basic output* approach satisfies both of the above. A detailed description of the I/O logic mechanisms and features can be found in [11], [12].

Boella et al. [1] described how a connectionist approach like neural networks can embed the different features of I/O logic: within this perspective, it is possible to use translation algorithms (like KBANN or CILP) to reproduce the mechanisms of I/O logic. In many examples of this paper, since we are dealing with normative reasoning, the consequents of the rules will be expressed using the O operator: for instance, $(getFine, O(payFine))$ represent the norm *If you are given a fine, you ought to pay it.*

2.3 Normative agent

In this paper we focus on modeling and reasoning about what a normative agent [2] is obliged or allowed to do in given states of the surrounding environment. Normative reasoning requires agents to deal with specific problems such as *dilemmas*, *exceptions* and *contrary to duties*.

Dilemmas: two obligations are said to be *contradictory* when they can not be accomplished together. A possible example of contradictory normas is the *dilemma*. This usually happens when an agent is subject to different normative codes (i.e. when an agent has to follow the *moral* and the *legal* code). Anyway it is outside the scope of this paper to discuss about how to overcome dilemmas, as we are focusing on how to use *priorities* to regulate *exceptions* and *contrary to duties*.

Priorities are used to give a partial ordering between norms. This is useful when, given two applicable norms, we always want one to preempt the other, for instance when dealing with *exceptions*.

We encode priorities among the norms by using *negation as failure* ($\sim$). Given two norms $R_1 = (A_1 \wedge A_3, \mathbf{O}(\beta_1))$ and $R_2 = (A_2 \wedge A_3, \mathbf{O}(\beta_2))$ and a priority relation $R_1 \succ R_2$ between them (such that the first norm has priority), we encode the priority relation by modifying the antecedent of the norm with lower priority. Specifically, we include in the antecedent of the norm with the lower priority the negation as failure of the literals in the antecedent of the higher prioritized norm that does not appear in the antecedent of the lower priority norm. We do so in order to ensure that, in a situation where both (unmodified) norms would be applicable, the newly inserted negation-as-failure atoms in the antecedent of the modified lower-prioritize rule evaluate to false and make the whole rule not applicable. Considering for example the two rules given above, we have to modify R_2. The only atom appearing in R_1's antecedent and not in R_2's antecedent is A_1, and therefore we introduce $\sim A_1$ as a conjunct in R_2's antecedent. After embedding the priority, the second rule becomes $R'_2 = (A_2 \wedge \sim A_1 \wedge A_3, \mathbf{O}(\beta_2))$. Note that in a potentially conflicting situation when A_1, A_2 and A_3 hold, R_1 and R_2 are applicable, but R'_2 is not, thus avoiding the conflict.

Exceptions occur when, due to particular circumstances, a norm should be followed instead of another one. Suppose that a norm $R_1 = (\alpha, \mathbf{O}(\beta))$ should be applied in all the situations containing α. For exceptional situations we consider an additional norm $R_2 = (\alpha \wedge \gamma, \mathbf{O}(\neg\beta))$. The latter norm should be applied in a subset of situations w.r.t. R_1: specifically all those when, in addition to α, also γ holds. We can call situations where both α and γ hold exceptional situations. In these exceptional situations both norms could be applied. This would produce two contrasting obligations: $\mathbf{O}(\beta)$ and $\mathbf{O}(\neg\beta)$. To avoid this we add the following priority relation: $R_2 \succ R_1$. Therefore we modify the antecedent of the norm with lower priority as described earlier. The result is a new norm $R'_1 = (\alpha \wedge \sim \gamma, \mathbf{O}(\beta))$, that would not be applied in the exceptional situations, avoiding the problem of contrasting obligations.

Contrary to Duties: an important property of norms is that they are soft constraints. Accordingly to this feature they can be violated. Contrary to duties provide additional obligations to be fulfilled when a violation occurs.

For example, consider a norm $R_1 = (\alpha, \mathbf{O}(\beta))$ that should be applied in all situations containing α and producing the obligation $\mathbf{O}(\beta)$. As mentioned, norms can be violated, therefore we can also define a norm that produces alternative obligations to be followed in case of a violation. Let this new norm be $R_2 = (\alpha \wedge \neg\beta, \mathbf{O}(\gamma))$. The latter norm contains in its antecedent both the antecedent of R_1 and the negation of its consequent. In this way it describes which should be the alternative obligation to $\mathbf{O}(\beta)$ in the case that it can not be achieved, in this example $\mathbf{O}(\gamma)$.

We use a priority relation between the two norms in order to avoid the generation of the obligation $\mathbf{O}(\beta)$ in case it is already known that it is not satisfiable. We add then the following priority relation $R_2 \succ R_1$ that modifies the first norm as follows: $R'_1 = (\alpha \wedge \sim \neg\beta, \mathbf{O}(\beta))$.

Permissions: an important distinction between *oughts* and *permissions* is that the latter will not be explicitly encoded in the neural network. In our approach we consider that something is permitted to the agent if not explicitly forbidden (note that we consider the ought of a negative literal as a prohibition). Due to this we consider that rules with a permission in their consequent implicitly have priority over the rules that forbid the same action. For example, consider two rules $R_1 = (A_1, \mathbf{P}(\beta_1))$, $R_2 = (A_2, \mathbf{O}(\neg\beta_1))$. The first rule permits β_1 and the second forbids it. In this case we assume the following priority relation $R_1 \succ R_2$ holds.

3. ARCHITECTURE AND CASE STUDY

Our goal is to allow the agent to learn from experience and take decisions which respect the norms she is subject to. Thus, the agent needs to know what is obligatory and forbidden according to norms (conditional rules) in any situation in real time. What is obligatory can eventually become an action of the agent, while what is forbidden inhibits such actions, like in agent architectures [6].

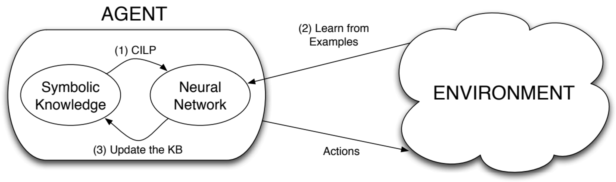


Figure 1: Normative agent architecture.

Rules may change: the normative environment changes over time so the agent should be flexible enough to adapt its behavior to the context using as information the instances of behaviors which have been considered illegal.

Figure 1 describes our approach. It starts from the symbolic knowledge-base (KB) of norms contained in the agent, transforming it into a neural network (NN) using an extension of the CILP algorithm (introduced below). The NN is structured as follows: input neurons of the network represent the state of the world (e.g., in the robocup domain, kickoff, have ball, etc.), while the output neurons represent the obligations of the agent, e.g., pass the ball (i.e. cooperate), minimize impact, etc., or the prohibitions, e.g., do not pass, do not score own goal, etc. The NN is used as part of the controller for the agent and, given its ability to learn, it is hoped to give the agent the required flexibility.

We then train the NN on instances of robocup match behaviors to adapt the agent to the current context. E.g., given a set of situations where the referee punishes an agent for kicking the ball backwards, we specify them as learning instances where there is the prohibition to kick the ball backwards. The NN can generalize the conditions under which this prohibition holds. To learn from behaviors which are regulated by norms, the NN must be able to cope with the peculiarities of normative reasoning.

In our tests we used a version of the RoboCup rules from the 2007 competition where, for simplicity, teams are composed of two players. To make things more interesting, in addition to those rules, we have added to the KB some norms representing the *coach*'s directions that regulate the behavior of the robots during the match.

Each rule is of the form IF α THEN β. The precondition α is a set of literals in conjunctive form while the postcondition β can be either an obligation or a permission concerning a single literal. Rules like IF $\top$ THEN O($\neg$*impact_opponent*) and IF *have_ball* $\wedge$ *opponent_approaching* THEN O(*pass*) contain obligations in their postconditions. Differently, a rule like IF *goalkeeper*$\wedge$*inside_own_area* THEN P(*use_hands*) contains a permission.

It is possible, however, that the environment requires the agent to adopt some sub-optimal behavior in circumstances when the optimal solution is not available. We use priorities to manage general and specific rules, creating a general-to-specific superiority relation and dealing with sub-optimal and exceptional situations. The two rules that compose an instance of contrary to duty are in the following configuration: the first one IF α THEN O(β) and IF $\neg\beta$ THEN O(γ); β represents the obligation to be fulfilled in an or-

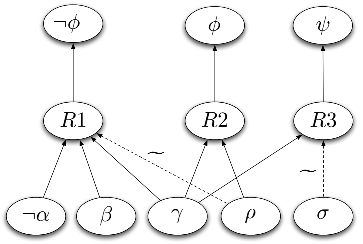


Figure 2: Example of I/O logic embedding in a NN

dinary situation α. If the agent is in a state of the world where β cannot be fulfilled, the second rule overcomes the first one through the use of priorities. For instance, IF $\top$ THEN O($\neg$*impact_opponent*) $\prec$ IF *impact_opponent* THEN O(*minimize_impact*). Intuitively, we use ($\prec$) such that $(y) \prec (x)$ means that, whenever the conclusion of rule (x) holds, the conclusion of (y) does not hold.

Figure 2 shows a neural network built from four rules: $R_1 = (\neg\alpha \wedge \beta \wedge \gamma, \mathbf{O}(\neg\phi))$, $R_2 = (\gamma \wedge \rho, \mathbf{O}(\phi))$, $R_3 = (\gamma, \mathbf{O}(\neg\psi))$ and the *permission rule* $R_4 = (\gamma \wedge \sigma, \mathbf{P}(\psi))$. In addition, a priority ordering $R_2 \succ R_1$ is expected to inhibit the activation of the first rule whenever the second rule applies. This priority is embedded within the rules as described earlier and, as a result, we obtain a new first rule: $R'_1 = (\neg\alpha \wedge \beta \wedge \gamma \wedge \sim \rho, \mathbf{O}(\neg\phi))$. Further, the implicit priority of R_4 over R_3 embeds in R_3 a negative literal obtaining a new rule, as follows: $R'_3 = (\gamma \wedge \sim \sigma, \mathbf{O}(\neg\psi))$. The neural network is built, then, from rules R'_1, R_2 and R'_3 (permission rules are not encoded in the network and are only used to define the priorities). Dotted lines in the figure indicate links with negative weighted which, in turn, implement the negation in the rules R'_1 and R'_3. Notice how input and output neurons in the network have a natural correspondence with inputs and outputs in I/O logic. Each hidden neuron represents a rule, e.g. R_1, and the network, sometimes called an AND/OR network, is supposed to compute conjunctions in its hidden layer and disjunctions in its output layer. In what follows, we detail the algorithm that achieves this translation and its proof of soundness w.r.t. an answer set semantics. Notice that, although the network is associated with a logic programming semantics, it has very naturally an input and output layer that make it appropriate, rather like I/O logic, for normative reasoning. This will be exemplified later.

4. NEURAL NETWORKS FOR NORMS

In this section we introduce a new approach for coding (a fragment of) I/O logic into a neural network. The main intuition behind this methodology is that, although logic programs do not capture the concepts of *inputs* and *outputs*, an extended logic program-based neural network does, on a purely structural level: inputs and outputs in I/O logics correspond to the input and output layers of the neural network.

Neural-symbolic algorithms (like CILP) provide a sound and complete translation of logic programs (LP) into a neural network (NN). Unfortunately, LP is not directly suitable for reasoning about normative systems (in particular about CTD and dilemmas). This is due to the fact that LP does not have an explicit representation of inputs.

A fact a in an LP could be mapped, at first sight, as the input of the NN, so to make rules like $a \rightarrow b$ fire to produce output b. At the same time, a should be also among the output of the network, due to identity property of the underlying logic: a follows from a. But this would require to implement identity property in the NN, making it more complicated.

CILP does not need to represent a fact as an initial input, thanks to transitivity property of logic, which is expressed by the fact that the NN is recurrent: every output neuron is connected to the corresponding input neuron.

If the fact a was directly represented as an output, it would not need to be represented as an explicit input, since the transitivity property allows to propagate output to input.

To minimize the structure of the network, CILP translates a fact a (representing the input to other rules) directly as an output a of the neural network and, given a rule like $a \rightarrow b$, to derive b as output, the output a becomes the "input" of the NN due to the fact that the NN is recurrent: every output becomes an input subsequently, rather than at the initial iteration. So in a sense the NN resulting from CILP given an LP returns always the same output after the network stabilizes, since it has no explicit input.

In normative reasoning, as captured by IO logic, the input does not become necessarily an output, since identity does not hold. The reason is that the output is interpreted as what is obligatory, thus, if a is in the input, it is not necessarily the case that a is obligatory as well. Differently from LP, what is in the input must be distinguished from the output: a fact a cannot be modeled as an output which becomes an input due to transitivity. As an example, the logic programs $P_1 = \{\emptyset\}$ and $P_2 = \{a \rightarrow b\}$ both have the empty set as model, this is because LP semantics do not reflect the meaning of the program rule. However, if we translate P_1 and P_2 with CILP we get two different networks, one with an empty set of input and output nodes and the other with a as the input note and b as output. The need to explicitly reason about inputs and outputs of rules in normative systems has been put forward by Makinson and van der Torre [11] in their Input-Output (I/O) Logic framework. In I/O logic, norms are represented as ordered pairs of formulas like (α, β), read as: if α is present in the current situation then β should be the case. These two formulae are also named correspondingly the *input* and the *output*, to make it clear that the input of the norm is the current situation and what is desirable for this situation is the output. A peculiarity of I/O logic (shared with conditional logics) is that it does not have (α, α) for any α (i.e. *identity* is not an axiom), while in LP we always have $\alpha \leftarrow \alpha$. This input/output perspective corresponds straightforwardly to the intuition behind a NN. However, to take advantage of the existing CILP algorithm and its proof of soundness we translate (a simplified) IO logic into LP to be processed by CILP without mapping the input into atoms translated as output. Rather the input is subsequently passed as input of the network producing an output representing what is obligatory, where some input appears in the output only if it is made obligatory by some rule.

In CILP output nodes are always connected to input nodes creating a recurrent network, to represent the transitivity property. In normative reasoning transitivity is not always accepted (since if you are obliged to do a and if a then you are obliged to b, does not imply that you are obliged to do b), thus the normative CILP thus extends CILP to account for the fact that certain outputs should not be connected to their corresponding inputs.

4.1 Mapping I/O Logic into Neural Networks

In this section, we first introduce a fragment of I/O logic, then we present an embedding of such fragment into extended logic programs and finally, we discuss how to represent priorities between rules within extended logic programs.

DEFINITION 1. *An* extended logic program *is a finite set of clauses of the form $L_0 \leftarrow L_1, \ldots, \sim L_n, \sim L_{n+1}, \ldots, \sim L_m$, where L_i $(0 \leq i \leq n)$ is a literal i.e., an atom or a classical negation of an atom denoted by $\neg$ and $\sim L_J$ $(n{+}1 \leq j \leq m)$ is called* default literal *where $\sim$ represents negation as failure.*

Given an extended logic program P we identify its *answer sets* [9] as $EXT(P)$.

DEFINITION 2 (I/O NORMATIVE CODE). *A normative code $\mathbf{G} = \langle \mathbb{O}, \mathbb{P}, \succ \rangle$ is composed by two sets of rules $r : (\alpha, \beta)$ and a preference relation $\succ$ among those rules. Rules in $\mathbb{O}$ are called* obligations*, while rules in $\mathbb{P}$ are* permissions. *Rules in $\mathbb{O}$ are of the type (α, β) where*

- *$\alpha = \alpha_1 \vee \ldots \vee \alpha_n$ is a propositional formula in disjunctive normal form i.e., α_i (for $0 \leq i \leq n$) is a conjunction of literals $(\neg a_{\alpha_{i1}} \wedge \ldots \wedge \neg a_{\alpha_{im}} \wedge a_{\alpha_{i(m+1)}} \wedge \ldots \wedge a_{\alpha_{1(m+p)}})$. Without loss of generality we assume that the first m literals are negative while the others $(m+p)-1$ are positive.*
- *$\beta = \neg b_{\beta_1} \wedge \ldots \wedge \neg b_{\beta_m} \wedge b_{\beta_{m+1}} \wedge \ldots \wedge b_{\beta_{m+p}}$ is a finite conjunction of literals.*

While rules in $\mathbb{P}$ are of type (α, l) where α is the same as for obligations but l is a literal.

As put forward in [4] the role of permissions is to undercut obligations. Informally, suppose to have a normative code $\mathbf{G}$ composed of two rules:

1. b is obligatory (i.e., $(\top, b) \in \mathbb{O}$).
2. If a holds, then $\neg b$ is permitted (i.e., $(a, \neg b) \in \mathbb{P}$).

We say that the rule $(a, \neg b)$ *has priority* over $(\top, b)$, i.e., b is obligatory as long as a does not hold, otherwise $\neg b$ is permitted and, therefore b is not obligatory anymore.

The semantics of such fragment of I/O is defined by the rules in Fig 3. $I(G)$ is the set of literals in the antecedent of rules in $\mathbf{G}$. The rules are a syntactical restriction of the those presented in [11].

The fact that we consider only I/O rules as defined in Definition 2 permits us to define a natural embedding of I/O rules and extended logic programs.

DEFINITION 3. *We define a function $\ulcorner\cdot\urcorner$ which embeds I/O logic rules into extended logic programs*

$$\lceil r : (\alpha_1 \vee \ldots \vee \alpha_n, \beta_1 \wedge \ldots \wedge \beta_m)\rceil =$$
$$\{r_{11} : (\lceil\beta_1\rceil_{out} \leftarrow \lceil\alpha_1\rceil_{in}); \ldots; r_{1m} : (\lceil\beta_m\rceil_{out} \leftarrow \lceil\alpha_1\rceil_{in})$$
$$; \ldots ;$$
$$r_{n1} : (\lceil\beta_1\rceil_{out} \leftarrow \lceil\alpha_n\rceil_{in}); \ldots; r_{nm} : (\lceil\beta_m\rceil_{out} \leftarrow \lceil\alpha_n\rceil_{in})\}$$

$$\lceil l_1 \wedge \ldots \wedge l_n\rceil_{in/out} = \lceil l_1\rceil_{in/out}, \ldots, \lceil l_n\rceil_{in/out}$$
$$\lceil a\rceil_{in} = in\_a \qquad \lceil a\rceil_{out} = out\_a$$
$$\lceil\neg a\rceil_{in} = \neg in\_a \qquad \lceil\neg a\rceil_{out} = \neg out\_a$$

we call rules r_{ij} as instances of r and we informally write $r_{ij} \in Ints(r)$.

Notice that the program resulting from the application of $\lceil\cdot\rceil$ has a unique model because it is negation-as-failure-free (NAF). Given a set of obligations $\mathbb{O}$, its closure $\mathbb{O}'$ under the rules of Fig. 3 exists and is finite.

LEMMA 1. *Given a set of obligations $\mathbb{O} = \{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\}$ and its closure $\mathbb{O}'$ under the rules defined in Fig. 3 we have*

$$\textit{If } (\alpha, \beta) \in \mathbb{O}' \textit{ then } \lceil\beta\rceil_{out} \in \mathcal{E} \in$$
$$EXT(\{\lceil(\alpha_1, \beta_1)\rceil; \ldots; \lceil(\alpha_n, \beta_n)\rceil\} \cup \lceil\alpha\rceil_{in})$$

PROOF. *First, we notice that $\mathcal{E}$ is unique (see Corollary 4.1). The* if *direction is trivial while the* only if *can be proved by showing that every application of the immediate consequence operator $\mathcal{T}$ (as defined in [9]) can be encoded into an application of the rules in Fig. 3.* □

We now show how to extend the preference relation $\succ$ w.r.t. rules generated with $\lceil\cdot\rceil$ as in Def. 3

DEFINITION 4. *Given a normative code $\mathbf{G} = \langle \mathbb{O}, \mathbb{P}, \succ \rangle$ we define a transformation $Tr_o(\cdot)$ such that $Tr_o(\mathbf{G}) = \langle \lceil\mathbb{O}\rceil, \mathbb{P}, \succ' \rangle$ where $\succ'$ is defined as follows:*

- *$t_{ij} \succ' t'_{i'j'}$, for all $t_{ij} \in Inst(t)$ and $t'_{i'j'} \in Inst(t')$ for $t, t' \in \mathbb{O}$ such that $t \succ t'$.*

For this reason, for a given normative code $Tr_o(\mathbf{G})$, we define a further transformation $Tr_p(\cdot)$ defined as follows

DEFINITION 5. *Given a normative code $\mathbf{G_o} = Tr_o(\mathbf{G}) = \langle \lceil\mathbb{O}\rceil, \mathbb{P}, \succ' \rangle$ we define $Tr_p(\mathbf{G_o}) = \langle \lceil\mathbb{O}\rceil, \mathbb{P}, \succ'' \rangle$, where $\succ''$ is defined as follows:*

- *For all $p : (\alpha, l) \in \mathbb{P}$, $p \succ'' t_{ij}$, for all $t_{ij} : (\alpha, \neg l) \in \lceil\mathbb{O}\rceil$*

We now discuss how to encode priorities between rules into extended logic programs [15].

DEFINITION 6. *Given a preference relation between r_i and r such that $r_i \succ r$ for $1 \leq i \leq j$,*
Replace the clause $r : L_1, ..., L_p \rightarrow L_{q+1}$ by clause $L_1, ..., L_p, \sim L^1_{p+1}, ..., \sim L^1_q, ..., \sim L^j_{p+1}, ..., \sim L^j_q \rightarrow L_{q+1}$,
where $r_{i(1 \leq i \leq j)} : L^i_{p+1}, ..., L^i_q \rightarrow L^i_{q+1}$;

EXAMPLE 1. *Suppose to have the following normative code $\mathbf{G} = \langle \{r : (a, \neg b \wedge c)\}, \{p : (d, b)\}, \{\} \rangle$, then $Tr_o(\mathbf{G}) = \{\langle r_{11} : (a, \neg b); r_{12} : (a, c)\}, \{p : (d, b)\}, \{\} \rangle$ and $Tr_p(Tr_o(\mathbf{G})) = \{\langle r_{11} : (a, \neg b); r_{12} : (a, c)\}, \{p : (d, b)\}, \{p \succ r_{11}\} \rangle$.*

Rules with permissions in the consequent, which are of the form $p_i : L_{i_1}; \ldots; L_{i_n}; L_{i_{n+1}}; \ldots; L_{i_m} \rightarrow L_{i_{m+1}}$ such that, for any other rule $r : L_{i_1}; \ldots; L_{i_n} \rightarrow \neg L_{i_{m+1}}$ (resulting from the application of $\lceil\mathbf{G}\rceil$) we impose $p_i \succ r$. The role of permission rules is to undercut (obligations rules) in $\lceil\mathbf{G}\rceil$ and will not be encoded into the symbolic neural network (every output encoded in the NN counts as an obligation, permission are not represented in the network but something is permitted if the contrary is not obligatory, see Section 4.2).

LEMMA 2. *Let $P_\succ = \{r_1, r_2, ..., r_n\}$ be an extended program with an explicit superiority relation $\succ$. Let P denote the translation of $P_\succ$ into a program without $\succ$. We have that $EXT(P_\succ) = EXT(P)$.*

$$\frac{(\alpha, \alpha_{o_1} \wedge \alpha_{o_2} \wedge \ldots \wedge \alpha_{o_n})}{(\alpha, \alpha_{o_2} \wedge \ldots \wedge \alpha_{o_n})}(WO) \quad \frac{(\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n, \beta)}{(\alpha_2 \vee \ldots \vee \alpha_n, \beta)}(WI) \quad \frac{(\alpha_{i_1} \wedge \ldots \wedge \alpha_{i_n}, \alpha_{o_1}) \qquad (\alpha_{i_1} \wedge \ldots \wedge \alpha_{i_n}, \alpha_{o_2})}{(\alpha_{i_1} \wedge \ldots \wedge \alpha_{i_n}, \alpha_{o_1} \wedge \alpha_{o_2})}(CO)$$

$$\frac{(\alpha_{i_1} \wedge \ldots \wedge \alpha_{i_n}, \gamma_{o_1}) \qquad (\beta_{i_1} \wedge \ldots \wedge \beta_{i_n}, \gamma_{o_1})}{((\alpha_{i_1} \wedge \ldots \wedge \alpha_{i_n}) \vee (\beta_{i_1} \ldots \beta_{i_n}), \gamma_{o_1})}(DI) \qquad \frac{(\alpha_{i_1} \wedge \ldots \wedge \alpha_{i_n}, \alpha_{o_1})}{(\alpha_{i_1} \wedge \ldots \wedge \alpha_{i_n} \wedge \beta_{i_1}, \alpha_{o_1})}(SI)_{\text{with } \beta_{i_1} \in I(G)}$$

Figure 3: Semantics for I/O Logic

We are interested in the translations above between $P_{\succ}$ and P because it is well-known that CILP networks will always settle down in the unique answer set of P provided P is well-behaved (i.e. locally stratified or acyclic or acceptable, see [7]). This result will be explored further in what follows.

4.2 The N-CILP algorithm

In this section we introduce the translation algorithm we have implemented in order to encode a normative code into a feed-forward NN (with semi-linear neurons), namely the Normative-CILP (N-CILP) algorithm. The proposed algorithm differs from standard CILP [7] in how priorities are encoded into the resulting neural network and does not connect input and output neurons that represent the same atom.

N-CILP

Given a normative code $\mathbf{G}$

1. $\mathbf{G}' = Tr_o(\mathbf{G}); G'' = Tr_p(\mathbf{G}')$
2. Apply the encoding of priorities as described in Definition 6 to $\mathbf{G}''$.
3. For each rule $R_k = \beta_{o_1} \leftarrow \alpha_{i_1}; \ldots; \alpha_{i_n}; \sim \alpha_{i_n+1}; \ldots; \sim \alpha_{i_m} \notin \mathbb{P}$.
   (a) For each literal α_{i_j} $(1 \leq j \leq m)$ in the input of the rule. If there is no input neuron labeled α_{i_j} in the input level, then add a neuron labeled α_{i_j} in the input layer.
   (b) Add a neuron labeled N_k in the hidden layer.
   (c) If there is no neuron labeled β_{o_1} in the output level, then add a neuron labeled β_{o_1} in the output layer.
   (d) For each literal α_{i_j} $(1 \leq j \leq n)$; connect the respective input neuron with the neuron labeled N_k in the hidden layer with a positive weighted arc.
   (e) layer with a negative weighted arc (the connections between these input neurons and the hidden neuron of the rule represents the priorities translated with the NAF).
   (f) Connect the neuron labeled N_i with the neuron in the output level labeled β_{o_1} with a positive weighted arc (each output in the rules is considered as a positive atom during the translation, this means that if we have a rule with a negative output $\neg\beta$, in the network we translate an output neuron labeled β' that has the same meaning of $\neg\beta$ but for the translation purpose can be treated as a positive output).

PROPOSITION 1. *For any normative code in the form of an extended logic program there exists a neural network obtained from the N-CILP translation algorithm such that the network computes the answer set semantics of the code.*

PROOF. *Def. 3.3 translates a normative code into an extended logic program having a single extension (or answer set). From Lemma 3.11, the program extended with a priority relation also has a single extension. In [7] it is shown that any extended logic program can be encoded into a neural network. N-CILP performs one such encoding using network weights as defined in [7]. Hence, N-CILP is sound. Since the program has a single extensions, the iterative recursive application of input-output patterns to the network will converge to this extension, which is identical to the unique answer set of the program, for any initial input.* □

5. EXPERIMENTAL RESULTS

The N-CILP algorithm was implemented as part of a simulator which is available online. In the simulator, the KB contains the rules that an agent knows. We assume that the priorities are embedded in the rules following the description used in the previous section. The KB is then read as input for the *N-CILP* translation which produces a standard NN for training. The network can be then trained within the simulator by backpropagation.

In this section, we describe the results of experiments carried out using the N-CILP simulator for network translation and training.

To evaluate the performance of the network, we use two distinct measures: *tot* and *part.*

$$tot = \frac{\sum_{i=1}^{n} I(\bigwedge_{j=1}^{k}(c_{ij} == o_{ij}))}{n}$$

$$part = \frac{\sum_{i=1}^{n}\sum_{j=1}^{k} I(c_{ij} == o_{ij})}{n * k}$$

where n refers to the cardinality of the test set, k is the number of output neurons in the network, o_{ij} is the value of the j-th output of the NN for the i-th test instance, c_{ij} is the true value (desired value) of the j-th literal for the i-th test instance, $I(\cdot)$ is the indicator, a function returning 1 if the argument is true and zero otherwise. The *tot* measure evaluates how many instances were processed entirely correctly, whle *part* considers the number of single output neurons correctly activated.

In our experiments we train the network using a *10fold cross validation.* We divide the initial data set of instances in ten distinct subsets. Each subset is then used as test set while the others are used together as training set. In this way the instances seen during training are left out of the testing phase, ten networks are trained and the results are averaged. The test-set performance provides us with an estimate of the network's generalization capability, i.e. its ability to predict the results (network output) for new instances (inputs), not seen during training. In all the experiments, we set the training parameters for the networks as follows: *learning rate*: 0.8, *momentum*: 0.3 and *training cycles*: 100. The reader is referred to [10] for the details of the backpropagation learning algorithm with momentum.

Non-symbolic approach comparison: we compare the learning capacity of a network built with N-CILP with a non-symbolic neural network. One of the well known issues in neural-network training is how to decide the number of neurons in the hidden layer. In the case of N-CILP, this number is given by the number of symbolic rules. We adopt the same number of hidden neurons for both networks, in order to avoid the risk of an unfair comparison with a randomly assessed topology for the non-symbolic network. The difference between the networks involved in this test lies in their connection weights. The neural network built with N-CILP sets its weights according to the rules in the KB. Instead, the non-symbolic network has its weights randomly initialized. One advantage of a network built with N-CILP is that even without any training, it is capable of correctly processing certain instances by applying the rules contained in the KB (if the rules are correct).

The network built with N-CILP has the head-start of a KB containing 20 rules. During the training phase, the network tries to learn 9 additional rules provided in the form of training instances (examples of input/output patterns). The non-symbolic network is provided with the same instances, including the instances for the initial 20 rules, but has to learn all the 29 rules using backpropagation.

The results from this little experiment show that the non-symbolic neural network is not able to achieve the same level of accuracy as the N-CILP network. For the non-symbolic network *tot* = 5.13% and *part* = 45.25%. For the N-CILP network *tot* = 5.38% and *part* = 49.19%. We can see that with the same knowledge provided as rules or instances, the networks achieve different results with the N-CILP network showing an improved performance.

Enhancing the knowledge base: the second experiment measures how the neural network performs by increasing the number of rules in the knowledge base. This test is important because the goal of a *Neural-Symbolic System*, is not only to construct a neural network capable to compute the same semantics as rule into the knowledge base. Another important objective is to exploit the learning capabilities of the neural networks, allowing the agent to increase the number of rules in its knowledge base from what it learned[7].

The test is done incrementally. From the full set of 29 rules, the experiment first step starts with a knowledge base containing 20 rules and tries to learn the remaining 9. Successively 2 rules are incrementally added into the initial knowledge base during each step. In this way the unknown rules that the network has to learn decreases by 2 each step. In example at the second step of the experiment the starting knowledge base contains 22 rules and the network tries to learn 7 rules during the training phase.

During each step the neural network is tested over instances where the full set of rules is applied. In this way the network continues to process using the rules already known, reducing the risk to forget them and in the meantime it tries to learn of the unknown rules.

The results of this experiment are shown in Figure 4. We can see that for the first two steps of the experiment the accuracies measured quite low. instead for the last two steps the performance of the neural network increases, reaching an accuracy peak of 98,01% for the *part* measure and 91,18%

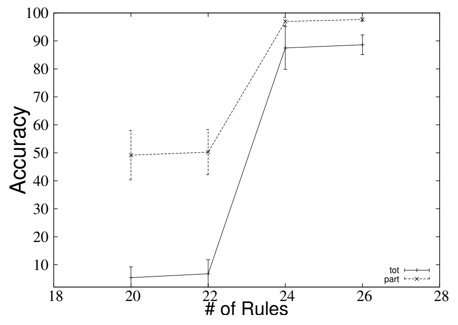


Figure 4: Accuracy of *tot* and *part* measures increasing the number of rules

for the *tot*.

From the experiment proposed we observed a direct correlation between the number of the rules in the starting knowledge base and the performance of the neural network. Another thing that can be noticed is that the smaller becomes the number of rules that the network does not know, w.r.t. the number of rules in the initial knowledge base can impact the performances of the network, also due to the fact that a network built from a larger knowledge base possesses more connections.

Learning Contrary to Duties: in this test we measure the capacity of a neural network built with N-CILP to learn new contrary to duties. In this case we use a starting knowledge base where the priority-based orderings regulating the contrary to duties were missing.

We tested the network on learning three different contrary to duties. The first refers to a situation where a robot player should never impact on an opponent. But if a collision route is inevitable, then the robot should make its best to minimize the impact. The second manages the situation where the robot is in physical contact with an opponent, which is forbidden by the RoboCup rulings. The robot should then try to terminate the contact. The third handles the situation where the robot is touching the ball with his hands, but he is not supposed to.

By removing the priority based orderings what is obtained is an incomplete system that produces, in similar situations, both the unfulfillable obligation and the relative obligation to handle the suboptimal situation that is being analyzed. What we expect from this test is that our approach is capable to learn from the examples, the priority based orderings that regulates the contrary to duties.

The neural network is trained with a set of instances that contain both normal situations and situations in which the contrary to duty is applied. The resulting network is tested with a test set containing sub-optimal situations, where an application of the contrary to duty is necessary. From the results of this test we verify that regarding the first contrary to duty, in the test set 95% of the instances were processed correctly and generating only the output obligation for the suboptimal situation that is what is desired on those situations. For the two other contrary to duties, we obtain an accuracy equal to 93% and 87% with their respective test sets.

Our approach is capable to learn contrary to duties not included in the construction of the neural network. This is a strength of the neural-symbolic architecture, that allows to avoid a total description of the investigated domain that could be, in some cases, very expensive and infeasible.

6. CONCLUSION

To the best of our knowledge this paper is the first to combine normative reasoning and learning with connectionist systems. Concerning the learning of normative systems in general, it is possible – as this paper also shows through the proposed translation of I/O logic into extended logic programs – to use a purely symbolic set-up. In the experiments proposed we see that a neural-symbolic approach has some advantages w.r.t. a pure connectionist one. This approach solves problems like the decision a priori of the NN size. From the results obtained in the tests, we empirically show that embedding previous knowledge in the NN increases its learning and processing performances. Notably, it should be possible to learn the kind of extended programs that we are considering here through the use of Inductive Logic Programming (ILP) [14] (or some adaptation of it to accommodate the use of negation, for example [16]). ILP has been used successfully in bioinformatics, but we are not aware of its application in normative systems. It would be interesting to compare and contrast the performance of the symbolic and connectionist approaches in the context of a real normative-systems application. Measurable criteria for comparison would include: accuracy, learning performance and noise tolerance indexes.
In this paper we chose to focus on the translation in one direction, as this can be used for tasks such as creating adaptable controllers. As future work, we'll be working on extensions of the tool to include an extraction module so it can be used when explicit explanations are required.
The system described thus far considers only one type of norms: the so called regulative norms, i.e., the norms prescribing the behavior of agents, in terms of what is obligatory, forbidden or permitted. Future work is also introducing constitutive rules besides regulative ones prescribing what is obligatory, forbidden or permitted. Constitutive rules provide a classification of reality in terms of the so called institutional facts, like marriages, licences, authorizations, institutions, etc. In the antecedents of regulative rules refer to the situation in which the norm should apply not only in terms of the brute facts (i.e., to the physical world) but also to institutional facts. Institutional facts are also inputs of constitutive rules meaning that differently than regulative rules, constitutive rules respect cumulative transitivity. Constitutive rules can be seen as a component whose output is fed as input to the component of regulative rules. The challenge is to study the interaction between the learning of the two components.

7. REFERENCES

[1] G. Boella, S. Colombo Tosatto, A. S. d'Avila Garcez, and V. Genovese. On the relationship between i-o logic and connectionism. In *13th International Workshop on Non-Monotonic Reasoning*, 2010.

[2] G. Boella, S. Colombo Tosatto, A. S. d'Avila Garcez, D. Ienco, V. Genovese, and L. van der Torre. Neural symbolic systems for normative agents. In *10th International Conference on Autonomous Agents and Multiagent Systems*, 2011.

[3] G. Boella, G. Pigozzi, and L. van der Torre. Normative framework for normative system change. In *8th Int. Joint Conf. on Autonomous Agents and Multiagent Systems AAMAS 2009*, pages 169–176. IFAAMAS, 2009.

[4] G. Boella and L. van der Torre. Permission and authorization in normative multiagent systems. In *Procs. of Int. Conf. on Artificial Intelligence and Law ICAIL*, pages 236–237, 2005.

[5] G. Boella and L. van der Torre. A game theoretic approach to contracts in multiagent systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 36(1):68–79, 2006.

[6] J. Broersen, M. Dastani, J. Hulstijn, and L. van der Torre. Goal generation in the BOID architecture. cognitive science quarterly. In *Cognitive Science Quarterly*, volume 2(3-4), pages 428–447, 2002.

[7] A. d'Avila Garcez, K. Broda, and D. Gabbay. *Neural-Symbolic Learning Systems: Foundations and Applications*. Perspectives in Neural Computing. Springer, 2002.

[8] A. S. d'Avila Garcez and G. Zaverucha. The connectionist inductive learning and logic programming system. *Applied Intelligence*, 11:59–77, July 1999.

[9] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[10] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[11] D. Makinson, L., and V. D. Torre. Input-output logics. *J. of Philosophical Logic*, 29:2000, 2000.

[12] D. Makinson and L. van der Torre. Constraints for input/output logics. *Journal of Philosophical Logic*, 30:155–185, 2001.

[13] E. Menegatti. Robocup soccer humanoid league rules and setup, 2007.

[14] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *J. Log. Program.*, 19/20:629–679, 1994.

[15] D. Nute. Defeasible logic. In D. Gabbay and J. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, pages 353–396. Oxford University Press, 1994.

[16] O. Ray. Automated abduction in scientific discovery. In *Model-Based Reasoning in Science, Technology, and Medicine*, volume 64 of *Studies in Computational Intelligence*, pages 103–116. Springer, 2007.

[17] S. Sen and S. Airiau. Emergence of norms through social learning. In *Procs. of the 20th International Joint Conference on Artificial Intelligence - IJCAI*, pages 1507–1512, 2007.

[18] Y. Shoham and M. Tennenholtz. On the emergence of social conventions: Modeling, analysis, and simulations. *Artif. Intell.*, 94(1-2):139–166, 1997.

[19] G. G. Towell and J. W. Shavlik. Knowledge-based artificial neural networks. *Artif. Intell.*, 70:119–165, October 1994.

On Supervising Agents in Situation-Determined ConGolog

Giuseppe De Giacomo
Sapienza Univ. Roma
Rome, Italy
degiacomo@dis.uniroma1.it

Yves Lespérance
York University
Toronto, ON, Canada
lesperan@cse.yorku.ca

Christian Muise
University of Toronto
Toronto, ON, Canada
cjmuise@cs.toronto.edu

ABSTRACT

We investigate agent supervision, a form of customization, which constrains the actions of an agent so as to enforce certain desired behavioral specifications. This is done in a setting based on the Situation Calculus and a variant of the ConGolog programming language which allows for nondeterminism, but requires the remainder of a program after the execution of an action to be determined by the resulting situation. Such programs can be fully characterized by the set of action sequences that they generate. Hence operations like intersection and difference become natural. The main results of the paper are a characterization of the maximally permissive supervisor that minimally constrains the agent so as to enforce the desired behavioral constraints when some agent actions are uncontrollable, and a sound and complete technique to execute the agent as constrained by such a supervisor.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods

General Terms

Languages, Theory, Verification

Keywords

Agent Reasoning::Knowledge representation, Agent theories, Models and Architectures::Logic-based approaches and methods, Agent Reasoning::Reasoning (single and multi-agent)

1. INTRODUCTION

There has been much work on *process customization*, where a generic process for performing a task or achieving a goal is customized to satisfy a client's constraints or preferences [9, 11, 16]. Process customization gained special momentum in the context of web service composition [17]. For example in [12], a generic process provides a wide range of alternative ways to perform a task. During customization, alternatives that violate the constraints are eliminated. Some parameters in the remaining alternatives may be restricted or instantiated so as to ensure that any execution of the customized process will satisfy the client's constraints. A related approach to service composition synthesizes an orchestrator that controls the execution of a set of available services to ensure that they realize a desired service [15, 1].

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

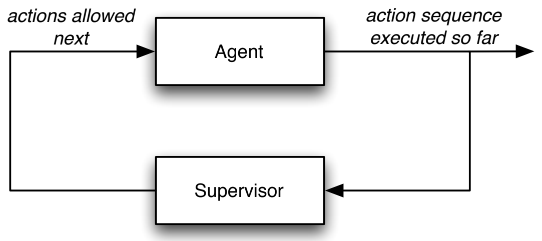


Figure 1: Supervised execution loop.

In this paper, we develop a framework for a similar type of process refinement that we call *supervised execution* (see Figure 1). We assume that we have a nondeterministic process that specifies the possible behaviors of an agent, and a second process that specifies the possible behaviors that a supervisor wants to allow (or alternatively, of the behaviors that it wants to rule out). For example, we could have an agent process representing a child and its possible behaviors, and a second process representing a babysitter that specifies the behaviors by the child that can be allowed. If the supervisor can control all the actions of the supervised agent, then it is straightforward to specify the behaviors that may result as a kind of synchronized concurrent execution of the agent and supervisor processes. A more interesting case arises when some agent actions are *uncontrollable*. For example, it may be impossible to prevent the child from getting muddy once he/she is allowed outside. In such circumstances, the supervisor may have to block some agent actions, not because they are undesirable in themselves (e.g., going outside), but because if they are allowed, the supervisor cannot prevent the agent from possibly performing some undesirable actions later on (e.g., getting muddy).

We follow previous work [12, 9] in assuming that processes are specified in a high level agent programming language defined in the Situation Calculus [14].[1] In fact, we define and use a restricted version of the ConGolog agent programming language [3] that we call Situation-Determined ConGolog (SDConGolog). In this version, following [4] all transitions involve performing an action (i.e., there are no transitions that merely perform a test). Moreover, nondeterminism is restricted so that the remaining program is a function of the action performed, i.e., given a program δ in a situation

[1]Clearly, there are applications where a declarative formalism is preferable, e.g., linear temporal logic (LTL), regular expressions over actions, or some type of business rules. However, there has been previous work on compiling such declarative specification languages into ConGolog, for instance [9], which handles an extended version of LTL interpreted over a finite horizon.

s, executing an action a (allowed by the program and the situation) results in a unique successor situation $s' = do(a, s)$ and a unique remaining program $\delta' = next(\delta, s, a)$, where $next$ is a function (formally defined later in this paper) that takes as arguments only δ, s, and a. This means that a run of such a program starting in a given situation can be taken to be simply a sequence of actions, as all the intermediate programs one goes through during the execution are functionally determined by the starting program and situation and the actions performed. Thus we can see a program in a situation as specifying a language formed by all the sequences of actions that are runs of the program in the situation. This allows us to define language theoretic notions such as union of languages, intersection, and difference/complementation in terms of operations on the corresponding programs, which has applications in many areas (e.g., programming by demonstration and programming by instruction [8], and plan recognition [6, 10]). We note that in [4], it is (implicitly) shown that any ConGolog program can be made situation-determined by recording nondeterministic choices as additional actions. Working with situation-determined programs also facilitates the formalization of supervision/customization. In particular it allows us to build on on the well-known Wonham and Ramadge framework for supervisory control of discrete event systems [13, 19, 2, 18].

Based on such a framework, we provide a general characterization of the *maximally permissive supervisor* that minimally constrains the actions of the agent specified in SDConGolog so as to enforce the desired behavioral specifications, showing its existence and uniqueness; secondly, we define a special program construct for *supervised execution* that takes the agent program and supervisor program, and executes them to obtain only runs allowed by the maximally permissive supervisor, showing its soundness and completeness.

The rest of the paper proceeds as follows. In the next section, we briefly review the Situation Calculus and the ConGolog agent programming language. In Section 3, we define SDConGolog, discuss its properties, and introduce some useful programming constructs and terminology. Then in Section 4, we develop our account of agent supervision, and define the maximally permissive supervisor and supervised execution. Finally in Section 5, we conclude the paper also discussing implementation.

2. PRELIMINARIES

The *situation calculus* is a logical language specifically designed for representing and reasoning about dynamically changing worlds [14]. All changes to the world are the result of *actions*, which are terms in the logic. We denote action variables by lower case letters a, action types by capital letters A, and action terms by α, possibly with subscripts. A possible world history is represented by a term called a *situation*. The constant S_0 is used to denote the initial situation where no actions have yet been performed. Sequences of actions are built using the function symbol *do*, such that $do(a, s)$ denotes the successor situation resulting from performing action a in situation s. Predicates and functions whose value varies from situation to situation are called *fluents*, and are denoted by symbols taking a situation term as their last argument (e.g., $Holding(x, s)$). Within the language, one can formulate action theories that describe how the world changes as the result of actions [14].

To represent and reason about complex actions or processes obtained by suitably executing atomic actions, various so-called *high-level programming languages* have been defined. Here we concentrate on (a fragment of) ConGolog that includes the following constructs:

α	atomic action
$\varphi?$	test for a condition
$\delta_1; \delta_2$	sequence
if φ **then** δ_1 **else** δ_2	conditional
while φ **do** δ	while loop
$\delta_1 \mid \delta_2$	nondeterministic branch
$\pi x.\delta$	nondeterministic choice of argument
δ^*	nondeterministic iteration
$\delta_1 \Vert \delta_2$	concurrency

In the above, α is an action term, possibly with parameters, and φ is situation-suppressed formula, that is, a formula in the language with all situation arguments in fluents suppressed. As usual, we denote by $\varphi[s]$ the situation calculus formula obtained from φ by restoring the situation argument s into all fluents in φ. Program $\delta_1|\delta_2$ allows for the nondeterministic choice between programs δ_1 and δ_2, while $\pi x.\delta$ executes program δ for *some* nondeterministic choice of a legal binding for variable x (observe that such a choice is, in general, unbounded). δ^* performs δ zero or more times. Program $\delta_1 \Vert \delta_2$ expresses the concurrent execution (interpreted as interleaving) of programs δ_1 and δ_2.

Formally, the semantics of ConGolog is specified in terms of single-step transitions, using the following two predicates [3]: *(i)* $Trans(\delta, s, \delta', s')$, which holds if one step of program δ in situation s may lead to situation s' with δ' remaining to be executed; and *(ii)* $Final(\delta, s)$, which holds if program δ may legally terminate in situation s. The definitions of $Trans$ and $Final$ we use are as in [4]; these are in fact the usual ones [3], except that the test construct $\varphi?$ does not yield any transition, but is final when satisfied. Thus, it is a *synchronous* version of the original test construct (it does not allow interleaving). As a consequence, in the version of ConGolog that we use, every transition involves the execution of an action (tests do not make transitions), i.e.,

$$\Sigma \cup \mathcal{C} \models \mathit{Trans}(\delta, s, \delta', s') \supset \exists a. s' = do(a, s).$$

Here and in the remainder, we use Σ to denote the foundational axioms of the situation calculus from [14] and $\mathcal{C}$ to denote the axioms defining the ConGolog programming language.

3. SD-PROGRAMS

We focus on a restricted class of ConGolog programs to describe processes, namely "situation-determined programs"; we call this class SDConGolog. A program δ is *situation-determined* in a situation s if for every sequence of transitions, the remaining program is determined by the resulting situation, i.e.,

$$\begin{array}{l} \mathit{SituationDetermined}(\delta, s) \doteq \forall s', \delta', \delta''. \\ \quad \mathit{Trans}^*(\delta, s, \delta', s') \land \mathit{Trans}^*(\delta, s, \delta'', s') \supset \delta' = \delta'', \end{array}$$

where *Trans** denotes the reflexive transitive closure of *Trans*. Thus, a (possibly partial) execution of a situation-determined program is uniquely determined by the sequence of actions it has produced. This is a key point. In general, the possible executions of a ConGolog program are characterized by sequences of configurations formed by the remaining program and the current situation. In contrast, the *execution of a SDConGolog program in a situation can be characterized in terms of a set of sequences (or language) of actions*. Such sequences correspond to situations reached from the situation where the program started.

For example, assuming for simplicity that all actions are executable in every situation, the ConGolog program $(a; b) \mid (a; c)$ is not situation-determined in situation S_0 as it can make a transition to a configuration $(b, do(a, S_0))$, where the situation is $do(a, S_0)$ and the remaining program is b, and it can also make a transition to a configuration $(c, do(a, S_0))$, where the situation is also $do(a, S_0)$

and the remaining program is instead c. It is impossible to determine what the remaining program is given only a situation, e.g. $do(a, S_0)$, reached along an execution. In contrast, the program $a; (b \mid c)$ is situation-determined in situation S_0. There is a unique remaining program $(b \mid c)$ in situation $do(a, S_0)$ (and similarly for the other reachable situations).

When we restrict our attention to situation-determined programs, we can use a simpler semantic specification for the language; instead of *Trans* we can use a $next$ (partial) function, where $next(\delta, a, s)$ returns the program that remains after δ does a transition involving action a in situation s (if δ is situation determined, such a remaining program must be unique). We will axiomatize the $next$ function so that it satisfies the following properties:

$$next(\delta, a, s) = \delta' \wedge \delta' \neq \bot \supset \mathit{Trans}(\delta, s, \delta', do(a, s)) \text{ (N1)}$$

$$\exists!\delta'.\mathit{Trans}(\delta, s, \delta', do(a, s)) \supset \\ \quad \forall\delta'.(\mathit{Trans}(\delta, s, \delta', do(a, s)) \supset next(\delta, a, s) = \delta') \text{ (N2)}$$

$$\neg\exists!\delta'.\mathit{Trans}(\delta, s, \delta', do(a, s)) \supset next(\delta, a, s) = \bot \text{ (N3)}$$

Here $\exists! x.\phi(x)$ means that there exists a unique x such that $\phi(x)$; this is defined in the usual way. $\bot$ is a special value that stands for "undefined". The function $next(\delta, a, s)$ is only defined when there is a unique remaining program after program δ does a transition involving the action a; if there is such a unique remaining program, then $next(\delta, a, s)$ denotes it.

We define the function $next$ inductively on the structure of programs using the following axioms.

Atomic action:

$$next(\alpha, a, s) = \begin{cases} nil \text{ if } Poss(a, s) \text{ and } \alpha = a \\ \bot \text{ otherwise} \end{cases}$$

Sequence: $next(\delta_1; \delta_2, a, s) =$

$$\begin{cases} next(\delta_1, a, s); \delta_2 \quad \text{if } next(\delta_1, a, s) \neq \bot \text{ and} \\ \qquad (\neg Final(\delta_1, s) \text{ or } next(\delta_2, a, s) = \bot) \\ next(\delta_2, a, s) \quad \text{if } \mathit{Final}(\delta_1, s) \text{ and } next(\delta_1, a, s) = \bot \\ \bot \quad \text{otherwise} \end{cases}$$

Conditional:

$$next(\textbf{if } \varphi \textbf{ then } \delta_1 \textbf{ else } \delta_2, a, s) = \begin{cases} next(\delta_1, a, s) \text{ if } \varphi[s] \\ next(\delta_2, a, s) \text{ if } \neg\varphi[s] \end{cases}$$

Loop:

$$next(\textbf{while } \varphi \textbf{ do } \delta, a, s) = \begin{cases} next(\delta, a, s); \textbf{while } \varphi \textbf{ do } \delta \\ \quad \text{if } \varphi[s] \text{ and } next(\delta, a, s) \neq \bot \\ \bot \text{ otherwise} \end{cases}$$

Nondeterministic branch:

$$next(\delta_1 | \delta_2, a, s) = \begin{cases} next(\delta_1, a, s) \text{ if } next(\delta_2, a, s) = \bot \text{ or} \\ \quad next(\delta_2, a, s) = next(\delta_1, a, s) \\ next(\delta_2, a, s) \text{ if } next(\delta_1, a, s) = \bot \\ \bot \text{ otherwise} \end{cases}$$

Nondeterministic choice of argument:[2]

$$next(\pi x.\delta, a, s) = \begin{cases} next(\delta^x_d, a, s) \text{ if } \exists! d.next(\delta^x_d, a, s) \neq \bot \\ \bot \text{ otherwise} \end{cases}$$

[2]Notice that d in δ^x_d depends on a and s. In particular d may be instantiated by the action a in s. Read on for an example.

Nondeterministic iteration:

$$next(\delta^*, a, s) = \begin{cases} next(\delta, a, s); \delta^* \quad \text{if } next(\delta, a, s) \neq \bot \\ \bot \text{ otherwise} \end{cases}$$

Interleaving concurrency: $next(\delta_1 \| \delta_2, a, s) =$

$$\begin{cases} next(\delta_1, a, s) \| \delta_2 \\ \quad \text{if } next(\delta_1, a, s) \neq \bot \text{ and } next(\delta_2, a, s) = \bot \\ \delta_1 \| next(\delta_2, a, s) \\ \quad \text{if } next(\delta_2, a, s) \neq \bot \text{ and } next(\delta_1, a, s) = \bot \\ \bot \text{ otherwise} \end{cases}$$

Test, empty program, undefined:

$$next(\varphi?, a, s) = \bot \quad next(nil, a, s) = \bot \quad next(\bot, a, s) = \bot$$

The undefined program is never *Final*: $\mathit{Final}(\bot, s) \equiv \texttt{false}$.

Let $\mathcal{C}^n$ be the set of ConGolog axioms extended with the above axioms specifying $next$ and $\mathit{Final}(\bot, s)$. It is easy to show that:

PROPOSITION 1. *N1, N2, and N3 are entailed by* $\Sigma \cup \mathcal{C}^n$.

Note in particular that as per N3, if the remaining program is not uniquely determined, then $next(\delta, a, s)$ is undefined. Notice that for situation-determined programs this will never happen, and if $next(\delta, a, s)$ returns $\bot$ it is because δ cannot make any transition using a in s:

COROLLARY 2.

$$\Sigma \cup \mathcal{C}^n \models \forall\delta, s.SituationDetermined(\delta, s) \supset \\ \forall a\,[(next(\delta, a, s) = \bot) \equiv (\neg\exists\delta'.\mathit{Trans}(\delta, s, \delta', do(a, s)))].$$

Let's look at an example. Imagine an agent specified by δ_{B1} below that can repeatedly pick an available object and repeatedly use it and then discard it, with the proviso that if during use the object breaks, the agent must repair it:

$$\delta_{B1} = [\pi\, x.Available(x)?; \\ \quad [use(x); (nil \mid [break(x); repair(x)])]^*; \\ \quad discard(x)]^*$$

We assume that there is a countably infinite number of available unbroken objects initially, that objects remain available until they are discarded, that available objects can be used if they are unbroken, and that objects are unbroken unless they break and are not repaired (this is straightforwardly axiomatized in the situation calculus). Notice that this program is situation-determined, though very nondeterministic.

Language theoretic operations on programs. We extend the SDConGolog language so as to close it with respect to language theoretic operations, such as *union*, *intersection* and *difference/complementation* of sets of sequences of actions. We can already see the nondeterministic branch construct as a union operator, and intersection and difference can be defined as follows.

Intersection/synchronous concurrency:

$$next(\delta_1 \,\&\, \delta_2, a, s) = \begin{cases} next(\delta_1, a, s) \,\&\, next(\delta_2, a, s) \\ \quad \text{if both are different from } \bot \\ \bot \text{ otherwise} \end{cases}$$

Difference: $next(\delta_1 - \delta_2, a, s) =$

$$\begin{cases} next(\delta_1, a, s) - next(\delta_2, a, s) \text{ if both are different from } \bot \\ next(\delta_1, a, s) \quad \text{if } next(\delta_2, a, s) = \bot \\ \bot \text{ if } next(\delta_1, a, s) = \bot \end{cases}$$

For these new constructs, *Final* is defined as follows:

$$\mathit{Final}(\delta_1 \,\&\, \delta_2, s) \equiv \mathit{Final}(\delta_1, s) \wedge \mathit{Final}(\delta_2, s)$$

$$\mathit{Final}(\delta_1 - \delta_2, s) \equiv \mathit{Final}(\delta_1, s) \wedge \neg\mathit{Final}(\delta_2, s)$$

We can express the *complement* of a program δ using difference as follows: $(\pi a.a)^* - \delta$.

It is easy to check that Proposition 1 and Corollary 2 also hold for programs involving these new constructs.[3]

As we will see later, synchronous concurrency can be used to constrain/customize a process. Difference can be used to prohibit certain process behaviors: $\delta_1 - \delta_2$ is the process where δ_1 is executed but δ_2 is not. To illustrate, consider an agent specified by program δ_{S1} that repeatedly picks an available object and does anything to it provided it is broken at most once before it is discarded:

$$\begin{aligned}\delta_{S1} = [&\pi\, x.Available(x)?;\\ &[\pi\, a.(a{-}(break(x) \mid discard(x)))]^*;\\ &(nil \mid (break(x)); [\pi\, a.(a{-}(break(x) \mid discard(x)))]^*);\\ &discard(x)]^*\end{aligned}$$

Sequences of actions generated by programs. We now characterize situation determined programs in terms of sequences of actions (runs) that may be performed from a given starting situation. This allows us to associate to such programs a language formed by such sequences of actions. Notice that in most cases not only the language will be infinite, but even the alphabet on which the language is defined will be infinite, since it is formed by all actions obtained by substituting values for parameters in the action types.

We start by extending the function $next$ to deal with sequences of actions. We assume sequences of actions to be inductively defined on their length as follows: *(i)* ϵ is the sequence of action of length 0; *(ii)* given a sequence of actions $\vec{a}$ of length n, and an action a the sequence $a\vec{a}$ is a sequence of actions of length $n+1$. (Notice that we are considering only finite sequences of actions in this way.)[4] When convenient, we will use the notation $\vec{a}\vec{b}$ to denote the sequence of actions formed by concatenating the two subsequences $\vec{a}$ and $\vec{b}$. As a special case we use also $\vec{a}a$ where a is an action.

The extension of the function $next$ to sequences of actions is a function $next^*(\delta, \vec{a}, s)$ that takes a program δ, a sequence of actions $\vec{a}$, and a situation s, and returns the remaining program δ' after executing δ in s producing the sequence of actions $\vec{a}$, defined by induction on the length of the sequence of actions as follows:

$$\begin{aligned}&next^*(\delta, \epsilon, s) = \delta\\ &next^*(\delta, a\vec{a}, s) = next^*(next(\delta, a, s), \vec{a}, do(a, s))\end{aligned}$$

where ϵ denotes the empty sequence. Note that if along $\vec{a}$ the program becomes $\bot$ then $next^*$ returns $\bot$ as well.

Runs. We define the set $\mathcal{RR}(\delta, s)$ of (partial) *runs* of a program δ in a situation s as the sequences of actions that can be produced by executing δ from s:[5]

$$\mathcal{RR}(\delta, s) = \{\vec{a} \mid next^*(\delta, \vec{a}, s) \neq \bot\}$$

Note that if $\vec{a} \in \mathcal{RR}(\delta, s)$, then all prefixes of $\vec{a}$ are in $\mathcal{RR}(\delta, s)$ as well.

[3]We may extend the definition of $Trans$ to the new constructs $\circ \in \{\&, -\}$ as follows $Trans(\delta_1 \circ \delta_2, s, \delta', s') \equiv \exists a.s = do(a, s) \wedge \delta' = next(\delta_1 \circ \delta_2, a, s) \neq \bot$.

[4]Notice that such sequences of actions have to be axiomatized in second-order logic in a standard way, similarly to situations. Also as an alternative, sequences of actions could also be characterized directly in terms of "difference" between situations. For sake of brevity, we leave out the formalization of sequences of actions here, and just assume that such sequences have been fully characterized.

[5]Here and in what follows, we use set notation for readability; if we wanted to be very formal, we could introduce $\mathcal{RR}$ as a defined predicate, and similarly for $\mathcal{CR}$, etc.

Complete runs. Notice that not all runs in $\mathcal{RR}$ reach eventually a final configuration. We are interested in distinguishing those runs that do. Hence we define the set of complete runs and that of their prefixes, called the good runs. We define the set $\mathcal{CR}(\delta, s)$ of *complete runs* of a program δ in a situation s as the sequences of actions that can be produced by executing δ from s until a *Final* configuration is reached:

$$\mathcal{CR}(\delta, s) = \{\vec{a} \mid Final(next^*(\delta, \vec{a}, s), do(\vec{a}, s))\}$$

As an alternative characterization, we observe that for every sequence of actions $\vec{a}$, we have $\vec{a} \in \mathcal{CR}(\delta, s)$ iff $Do(\delta, s, do(\vec{a}, s))$ using the usual terminology of [14, 3].

Good runs. We define the set $\mathcal{GR}(\delta, s)$ of *good runs* of a program δ in a situation s as the sequences of actions that can be produced by executing δ from s which can be *extended* until a *Final* configuration is reached:

$$\mathcal{GR}(\delta, s) = \{\vec{a} \mid \exists\vec{b}.Final(next^*(\delta, \vec{a}\vec{b}, s), do(\vec{a}\vec{b}, s))\}$$

In other words $\vec{a} \in \mathcal{GR}(\delta, s)$ if $\vec{a}$ is a prefix of a sequence of action $\vec{a'} = \vec{a}\vec{b}$ such that $\vec{a'} \in \mathcal{CR}(\delta, s)$.

$\mathcal{RR}(\delta, s)$, $\mathcal{CR}(\delta, s)$, and $\mathcal{GR}(\delta, s)$ can be considered as three languages (of sequences of actions) generated by the program δ in s. This allows us to apply language theoretic notions to situation-determined programs, and we exploit this possibility for studying supervision.

Before turning to that, let's make a few observations. First, it is easy to see that $\mathcal{CR}(\delta, s) \subseteq \mathcal{GR}(\delta, s) \subseteq \mathcal{RR}(\delta, s)$, i.e., complete runs are good runs, and good runs are indeed runs. Moreover, $\mathcal{CR}(\delta, s) = \mathcal{CR}(\delta', s)$ implies $\mathcal{GR}(\delta, s) = \mathcal{GR}(\delta', s)$, i.e., if two programs in a situation have the same complete runs, then they also have the same good runs; however they may still differ in their sets of non-good runs, since $\mathcal{CR}(\delta, s) = \mathcal{CR}(\delta', s)$ does not imply $\mathcal{RR}(\delta, s) = \mathcal{RR}(\delta', s)$. We say that a program δ in s is *non-blocking* iff $\mathcal{RR}(\delta, s) = \mathcal{GR}(\delta, s)$, i.e., if all runs of the program δ in s can be extended to runs that reach a *Final* configuration.

Search construct. Interestingly in the literature on Situation Calculus based programs, a special construct $\Sigma(\delta)$ was introduced to ensure that the only actions produced by a program δ are those that eventually lead to a final state. This is the so called "search construct" [5].

We can add such a construct to SDConGolog, and use it to generate only good runs. The search construct Σ is characterized in terms of $next$ as follows:

$$next(\Sigma(\delta), a, s) = \begin{cases}\Sigma(next(\delta, a, s)) & \text{if there exists } \vec{a} \text{ s.t.}\\ \quad Final(next^*(\delta, a\vec{a}, s)) & \\ \bot & \text{otherwise}\end{cases}$$

$$Final(\Sigma(\delta), s) \equiv Final(\delta, s).$$

Intuitively, $next(\Sigma(\delta), a, s)$ does lookahead to ensure that action a is in a good run of δ in s, otherwise it returns $\bot$.

Notice that: *(i)* $\mathcal{RR}(\Sigma(\delta), s) = \mathcal{GR}(\Sigma(\delta), s)$, i.e., under the search construct all programs are non-blocking; *(ii)* $\mathcal{RR}(\Sigma(\delta), s) = \mathcal{GR}(\delta, s)$, i.e., $\Sigma(\delta)$ produces exactly the good runs of δ; *(iii)* $\mathcal{CR}(\Sigma(\delta), s) = \mathcal{CR}(\delta, s)$, i.e., $\Sigma(\delta)$ and δ produce exactly the same set of complete runs. Thus $\Sigma(\delta)$ trims the behavior of δ by eliminating all those runs that do not lead to a *Final* configuration.

Note also that if a program is *non-blocking* in s, then $\mathcal{RR}(\Sigma(\delta), s) = \mathcal{RR}(\delta, s)$, in which case there is no point in using the search construct. Finally, we have that: $\mathcal{CR}(\delta, s) = \mathcal{CR}(\delta', s)$ implies $\mathcal{RR}(\Sigma(\delta), s) = \mathcal{RR}(\Sigma(\delta'), s)$, i.e., if two programs have the same complete runs, then under the search construct they have exactly the same runs.

4. SUPERVISION

Let us assume that we have two agents: an agent B with behavior represented by the program δ_B and a supervisor S with behavior represented by δ_S. While both are represented by programs, the roles of the two agents are quite distinct. The first is an agent B that acts freely within its space of deliberation represented by δ_B. The second, S, is supervising B so that as B acts, it remains within the behavior permitted by S. This role makes the program δ_S act as a specification of allowed behaviors for agent B.[6]

The behavior of B under the supervision of S is constrained so that at any point B can execute an action in its original behavior, only if such an action is also permitted in S's behavior. Using the synchronous concurrency operator, this can be expressed simply as:

$$\delta_B \ \&\ \delta_S.$$

Note that unless $\delta_B \ \&\ \delta_S$ happens to be non-blocking, it may get stuck in dead end configurations. To avoid this, we need to apply the search construct, getting $\Sigma(\delta_B \ \&\ \delta_S)$. In general, the use of the search construct to avoid blocking, is always needed in the development below.

We can use the example programs presented earlier to illustrate. The execution of δ_{B1} under the supervision of δ_{S1} is simply δ_{B1} & δ_{S1} (assuming all actions are controllable). It is straightforward to show that the resulting behavior is to repeatedly pick an available object and use it as long as one likes, breaking it at most once, and repairing it whenever it breaks, before discarding it. It can be shown that the set of partial/complete runs of δ_{B1} & δ_{S1} is exactly that of:

$$\begin{array}{l}[\pi\, x.Available(x)?;\\ \quad use(x)^*;\\ \quad [nil \mid (break(x); repair(x); use(x)^*)];\\ \quad discard(x)]^*\end{array}$$

Uncontrollable actions. In the above, we implicitly assumed that all actions of agent B could be controlled by the supervisor S. This is often too strong an assumption, e.g., once we let a child out in a garden after rain, there is nothing we can do to prevent her/him from getting muddy. We now want to deal with such cases.

To do this, we follow the general approach of the well-known Wonham and Ramadge (W&R) framework for supervisory control of discrete event systems [13, 19, 2, 18], suitably extended to deal with rich languages such as those generated by SDConGolog programs.

We start by distinguishing between actions that are *controllable* by the supervisor and actions that are *uncontrollable*. The supervisor can block the execution of the controllable actions, but cannot prevent the supervised agent from executing the uncontrollable ones.

To characterize the uncontrollable actions in the situation calculus, we use a special fluent $A_u(a_u, s)$, which we call an *action filter*, that expresses that action a_u is uncontrollable in situation s.

[6]Note that, because of these different roles, one may want to assume that all configurations generated by (δ_S, s) are $Final$, so that we leave B unconstrained on when it may terminate. This amounts to requiring the following property to hold: $\mathcal{CR}(\delta_S, s) = \mathcal{GR}(\delta_S, s) = \mathcal{RR}(\delta_S, s)$. While reasonable, for the technical development below, we do not need to rely on this assumption.

Notice that, unlike in the W&R framework, we allow controllability to be *context dependent* by allowing an arbitrary specification of the fluent $A_u(a_u, s)$ in the situation calculus.

While we would like the supervisor S to constrain agent B so that δ_B & δ_S is executed, in reality, since S cannot prevent uncontrollable actions, S can only constrain B on the controllable actions. When this is sufficient, we say that the supervison specification δ_S is "controllable". Technically, following again W&R, this can be captured by saying that the *supervision specification δ_S is controllable wrt δ_B in situation s* iff:

$$\begin{array}{l}\forall \vec{a}a_u.\vec{a} \in \mathcal{GR}(\delta_S, s) \text{ and } A_u(a_u, do(\vec{a}, s)) \text{ implies}\\ \quad \text{if } \vec{a}a_u \in \mathcal{GR}(\delta_B, s) \text{ then } \vec{a}a_u \in \mathcal{GR}(\delta_S, s).\end{array}$$

What this says is that if we postfix a good run $\vec{a}$ for S with an uncontrollable action a_u that is good for B (and so $\vec{a}$ must be good for B as well), then this uncontrollable action a_u must also be good for S. By the way, notice that $\vec{a}a_u \in \mathcal{GR}(\delta_B, s)$ and $\vec{a}a_u \in \mathcal{GR}(\delta_S, s)$ together imply that $\vec{a}a_u \in \mathcal{GR}(\delta_B \ \&\ \delta_S, s)$.

What about if such a property does not hold? We can take two orthogonal approaches: *(i)* simply relax δ_S so that it places no constraints on the uncontrollable actions; *(ii)* require that δ_S be indeed enforced, but also disallow those runs that prevent δ_S from being controllable. We look at both approaches below.

Relaxed supervision. To define relaxed supervision we first need to introduce two operations on programs: *projection* and, based on it, *relaxation*. The *projection operation* takes a program and an *action filter* A_u, and projects all the actions that satisfy the action filter (e.g., are uncontrollable), out of the execution. To do this, projection substitutes each occurrence of an atomic action term α_i by a conditional statement that replaces it with the trivial test `true`? when $A_u(\alpha_i)$ holds in the current situation, that is:

$$\begin{array}{l}pj(\delta, A_u) = \delta^{\alpha_i}_{\textbf{if } A_u(\alpha_i) \textbf{ then } \texttt{true?} \textbf{ else } \alpha_i}\\ \text{for every occurrence of an action term } \alpha_i \text{ in } \delta.\end{array}$$

(Recall that such a test does not perform any transition in our variant of ConGolog.)

The *relaxation operation* on δ wrt $A_u(a, s)$ is as follows:

$$rl(\delta, A_u) = pj(\delta, A_u) \| (\pi a.A_u(a)?; a)^*.$$

In other words, we project out the actions in A_u from δ and run the resulting program concurrently with one that picks (uncontrollable) actions filtered by A_u and executes them. The resulting program no longer constrains the occurrence of actions from A_u in any way. In fact, notice that the remaining program of $(\pi a.A_u(a)?; a)^*$ after the execution of an (uncontrollable) filtered action is $(\pi a.A_u(a)?; a)^*$ itself, and that such a program is always *Final*.

Now we are ready to define *relaxed supervision*. Let us consider a supervisor S with supervision specitfication δ_S for agent B with behavior δ_B. Let the action filter $A_u(a_u, s)$ specify the uncontrollable actions. Then the *relaxed supervision* of δ_S (for $A_u(a_u, s)$) in s is the relaxation of δ_S so as that it allows every uncontrollable action, namely: $rl(\delta_S, A_u)$. So we can characterize the behavior of B under the relaxed supervision of S as:

$$\delta_B \ \&\ rl(\delta_S, A_u).$$

The following properties are immediate consequences of the definitions:

PROPOSITION 3. *The relaxed supervision $rl(\delta_S, A_u)$ is controllable wrt δ_B in situation s.*

PROPOSITION 4. $\mathcal{CR}(\delta_B \ \&\ \delta_S, s) \subseteq \mathcal{CR}(\delta_B \ \&\ rl(\delta_S, A_u), s)$.

PROPOSITION 5. *If $\mathcal{CR}(\delta_B \ \& \ rl(\delta_S, A_u), s) \subseteq \mathcal{CR}(\delta_B \ \& \ \delta_S, s)$, then δ_S is controllable wrt δ_B in situation s.*

Notice that, the first one is what we wanted. But the second one says that $rl(\delta_S, A_u)$ may indeed by more permissive than δ_S: some complete runs that are disallowed in δ_S may be permitted by its relaxation $rl(\delta_S, A_u)$. This is often not acceptable. Proposition 5 says that when the converse of Proposition 4 holds, we have that the original supervision δ_S is indeed controllable wrt δ_B in situation s. Notice however that even if δ_S is controllable wrt δ_B in situation s, it may still be the case that $\mathcal{CR}(\delta_B \ \& \ rl(\delta_S, A_u), s) \subset \mathcal{CR}(\delta_B \ \& \ \delta_S, s)$.

Maximally permissive supervisor. Next we study a more interesting, more conservative approach: we require the supervision specification δ_S to be fulfilled, and for getting controllability we further restrict the specification if needed. This is the classical approach adopted in the W&R framework, and indeed, we are able to show that the key result of the W&R framework is preserved in our generalized setting: there is, in principle, a unique maximally permissive way of restricting the supervision specification so that it still fulfills δ_S while being controllable. We call the resulting supervisor the *maximally permissive supervisor*.

To phrase this result in our setting, however, we need to augment our programming language with a new construct $\mathbf{set}(E)$ that takes an arbitrary set of sequences of actions E and makes it a program. For such a construct *next* and *Final* are defined as follows:

$$next(\mathbf{set}(E), a, s) = \begin{cases} \mathbf{set}(E') \ \text{ with } E' = \{\vec{a} \mid a\vec{a} \in E\} \\ \qquad \text{if } E' \neq \emptyset \\ \bot \ \text{ if } E' = \emptyset \end{cases}$$
$$Final(\mathbf{set}(E), s) \equiv (\epsilon \in E)$$

Thus $\mathbf{set}(E)$ can be executed to produce any of the sequences of actions in E.

Notice that for every program δ and situation s, we can define $E_\delta = \mathcal{CR}(\delta, s)$ such that $\mathcal{CR}(\mathbf{set}(E_\delta), s) = \mathcal{CR}(\delta, s)$. The converse does not hold in general, i.e., there are programs $\mathbf{set}(E)$ such that for all programs δ, not involving the $\mathbf{set}(\cdot)$ construct, $\mathcal{CR}(\mathbf{set}(E_\delta), s) \neq \mathcal{CR}(\delta, s)$. That is, the syntactic restrictions in SDConGolog may not allow us to represent some possible sets of sequences of actions.[7]

With the $\mathbf{set}(E)$ construct at hand, following [19], we may define the *maximally permissive supervisor* $mps(\delta_B, \delta_S, s)$ of the agent behavior δ_B which fulfills the supervision specification δ_S in situation s, as:

$$\begin{aligned} mps(\delta_B, \delta_S, s) = \mathbf{set}(\textstyle\bigcup_{E \in \mathcal{E}} E) \ \text{ where} \\ \mathcal{E} = \{E \mid E \subseteq \mathcal{CR}(\delta_B \ \& \ \delta_S, s) \\ \text{and } \mathbf{set}(E) \textit{ is controllable wrt } \delta_B \textit{ in } s\} \end{aligned}$$

Intuitively mps denotes the maximal set of runs that are effectively allowable by a supervisor that fulfills the specification δ_S, and which can be left to the arbitrary decisions of the agent behaving as δ_B on the uncontrollable actions. A quite interesting result is that, even in the general setting we are presenting, such a maximally permissive supervisor always *exists* and is *unique*. Indeed, we can show:

THEOREM 6. *For the maximally permissive supervisor $mps(\delta_B, \delta_S, s)$ the following properties hold:*

1. *$mps(\delta_B, \delta_S, s)$ always exists and is unique;*

2. *$mps(\delta_B, \delta_S, s)$ is controllable wrt δ_B in s;*

3. *For every possible controllable supervision specification $\hat{\delta}_S$ for δ_B in s such that $\mathcal{CR}(\delta_B \ \& \ \hat{\delta}_S, s) \subseteq \mathcal{CR}(\delta_B \ \& \ \delta_S, s)$, we have that $\mathcal{CR}(\delta_B \ \& \ \hat{\delta}_S, s) \subseteq \mathcal{CR}(\delta_B \ \& \ mps(\delta_B, \delta_S, s), s)$.*

PROOF. We prove the three claims separately.

Claim 1. It follows directly from the fact $\mathbf{set}(\emptyset)$ satisfies the conditions to be included in $mps(\delta_B, \delta_S, s)$.

Claim 2. It suffices to show that $\forall \vec{a} a_u . \vec{a} \in \mathcal{GR}(\delta_B \ \& \ mps(\delta_B, \delta_S, s), s)$ and $A_u(a_u, do(\vec{a}, s))$ we have that if $\vec{a} a_u \in \mathcal{GR}(\delta_B, s)$ then $\vec{a} a_u \in \mathcal{GR}(mps(\delta_B, \delta_S, s), s)$. Indeed, if $\vec{a} \in \mathcal{GR}(\delta_B \ \& \ mps(\delta_B, \delta_S, s), s)$ then there is an controllable supervision specification $\mathbf{set}(E)$ such that $\vec{a} \in \mathcal{GR}(\delta_B \ \& \ \mathbf{set}(E), \delta_S, s), s)$. $\mathbf{set}(E)$ being controllable wrt δ_B in s, if $\vec{a} a_u \in \mathcal{GR}(\delta_B, s)$ then $\vec{a} a_u \in \mathcal{GR}(\mathbf{set}(E), s)$, but then $\vec{a} a_u \in \mathcal{GR}(mps(\delta_B, \delta_S, s), s)$.

Claim 3. It follows immediately from the definition of $mps(\delta_B, \delta_S, s)$, by noticing that $\mathcal{CR}(\delta_B \ \& \ \hat{\delta}_S, s) = \mathcal{CR}(\delta_B \ \& \ \mathbf{set}(E_{\hat{\delta}_S}), s)$, and observing that $mps(\delta_B, \delta_S, s)$ is essentially the union of such controllable $\mathbf{set}(E_{\hat{\delta}_S})$. □

Returning to our running example, if we assume that the *break* action is uncontrollable (and the others are controllable), the supervisor $S1$ can only ensure that its constraints are satisfied if it forces $B1$ to discard an object as soon as it is broken and repaired. This is what we get as maximally permissive supervisor $mps(\delta_{B1}, \delta_{S1}, S_0)$, whose set of runs can be shown to be exactly that of:

$$\begin{aligned} &[\pi\, x. Available(x)?; \\ &\quad use(x)^*; \\ &\quad [nil \mid (break(x); repair(x))]; \\ &\quad discard(x)]^* \end{aligned}$$

By the way, notice that $(\delta_{B1} \ \& \ rl(\delta_{S1}, A_u))$ instead is completely ineffective since it has exactly the runs of δ_{B1}.

Unfortunately, in general, $mps(\delta_B, \delta_S, s)$ requires the use of the program construct $\mathbf{set}(E)$ which is mostly of theoretical interest. For this reason the above characterization remains essentially mathematical. So next, we develop a new construct for execution of programs under maximally permissive supervision, giving up on precomputing the maximally permissive supervision specification a priori and instead directly computing it online while the agent is operating.

Maximally permissive supervised execution. To capture the notion of maximally permissive execution of agent B with behavior δ_B under the supervision of S with behavior δ_S in situation s, we introduce a special version of the synchronous concurrency construct that takes into account the fact the some actions are uncontrollable. Without loss of generality, we assume that δ_B and δ_S both start with a common controllable action (if not, it is trivial to add a dummy action in front of both so as to fullfil the requirement). Then, we characterize the construct through *next* and *Final* as follows:

[7]Obviously there are certain sets that can be expressed directly in SDConGolog , e.g., when E is finite. However notice that in the general case the object domain may be infinite, and $\mathbf{set}(E)$ may not be representable as a finitary SDConGolog program.

$$
next(\delta_B \mathbin{\&_{A_u}} \delta_S, a, s) =
\begin{cases}
next(\delta_B, a, s) \mathbin{\&_{A_u}} next(\delta_S, a, s) \quad \text{if} \\
\quad next(\delta_B, a, s) \neq \bot \text{ and } next(\delta_S, a, s) \neq \bot \text{ and} \\
\quad \text{if } \neg A_u(a, s) \text{, then} \\
\quad\quad \text{for all } \vec{a_u} \text{ such that } A_u(\vec{a_u}, do(a, s)) \\
\quad\quad\quad \text{if } next^*(\Sigma(\delta_B), a\vec{a_u}, s) \neq \bot, \\
\quad\quad\quad \text{then } next^*(\Sigma(\delta_S), a\vec{a_u}, s) \neq \bot \\
\bot \quad \text{otherwise}
\end{cases}
$$

where $A_u(\vec{a_u}, s)$, meaning that action sequence $\vec{a_u}$ is uncontrollable in situation s, is inductively defined on the length of $\vec{a_u}$ as the smallest predicate such that: *(i)* $A_u(\epsilon, s) \equiv \texttt{true}$; *(ii)* $A_u(a_u\vec{a_u}, s) \equiv A_u(a_u, s) \wedge A_u(\vec{a_u}, do(a_u, s))$. Thus, the maximally permissive supervised execution of δ_B for the specification δ_S is allowed to perform action a in situation s if a is allowed by both δ_B and δ_S and moreover, if a is controllable, then for every sequence of uncontrollable actions $\vec{a_u}$, if $\vec{a_u}$ may be performed by δ_B right after a on one of its complete runs, then it must also be allowed by δ_S (on one of its complete runs). Essentially, an action a by the agent must be forbidden if it can be followed by some sequence of uncontrollable actions that violates the specification.

Final for the new construct is as follows:

$$Final(\delta_B \mathbin{\&_{A_u}} \delta_S, s) \equiv Final(\delta_B, s) \wedge Final(\delta_S, s).$$

This new construct captures exactly the maximally permissive supervisor; indeed the theorem below shows the *correctness of maximally permissive supervised execution*:

THEOREM 7.

$$\mathcal{CR}(\delta_B \mathbin{\&_{A_u}} \delta_S, s) = \mathcal{CR}(\delta_B \mathbin{\&} mps(\delta_B, \delta_S, s), s).$$

PROOF. We start by showing: $\mathcal{CR}(\delta_B \mathbin{\&_{A_u}} \delta_S, s) \subseteq \mathcal{CR}(\delta_B \mathbin{\&} mps(\delta_B, \delta_S, s), s)$. It suffices to show that $\delta_B \mathbin{\&_{A_u}} \delta_S$ is controllable for δ_B in s. Indeed, if this is the case, by considering that $\delta_B \mathbin{\&} mps(\delta_B, \delta_S, s)$ is the largest controllable supervisor for δ_B in s, and that $\mathcal{RR}(\delta_B \mathbin{\&} (\delta_B \mathbin{\&_{A_u}} \delta_S), s) = \mathcal{RR}(\delta_B \mathbin{\&_{A_u}} \delta_S, s)$, we get the thesis.

So we have to show that: $\forall \vec{a}a_u.\vec{a} \in \mathcal{GR}(\delta_B \mathbin{\&_{A_u}} \delta_S, s)$ and $A_u(a_u, do(\vec{a}, s))$ we have that if $\vec{a}a_u \in \mathcal{GR}(\delta_B, s)$ then $\vec{a}a_u \in \mathcal{GR}(\delta_B \mathbin{\&_{A_u}} \delta_S, s)$.

Since, wlog we assume that δ_B and δ_S started with a common controllable action, we can write $\vec{a} = \vec{a'}a_c\vec{a_u}$, where $\neg A_u(a_c, do(\vec{a'}, s))$ and $A_u(\vec{a_u}, do(\vec{a'}a_c, s))$ holds. Let $\delta'_B = next^*(\delta_B, \vec{a'}, s)$, $\delta'_S = next^*(\delta_S, \vec{a'}, s)$, and $s' = do(\vec{a'}, s)$. By the fact that $\vec{a'}a_c\vec{a_u} \in \mathcal{GR}(\delta_B \mathbin{\&_{A_u}} \delta_S, s)$ we know that $next(\delta'_B \mathbin{\&_{A_u}} \delta'_S, do(a_c, s')) \neq \bot$. But then, by de definition of $next$, we have that for all $\vec{b_u}$ such that $A_u(\vec{b_u}, s')$ if $\vec{b_u} \in \mathcal{GR}(\delta'_B, do(a_c, s'))$ then $\vec{b_u} \in \mathcal{GR}(\delta'_S, do(a_c, s'))$. In particular this holds for $\vec{b_u} = \vec{a_u}a_u$. Hence we have that if $\vec{a}a_u \in \mathcal{GR}(\delta_B, s)$ then $\vec{a}a_u \in \mathcal{GR}(\delta_S, s)$.

Next we prove: $\mathcal{CR}(\delta_B \mathbin{\&} mps(\delta_B, \delta_S, s), s) \subseteq \mathcal{CR}(\delta_B \mathbin{\&_{A_u}} \delta_S, s)$. Suppose not. Then there exist a complete run $\vec{a}$ such that $\vec{a} \in \mathcal{CR}(\delta_B \mathbin{\&} mps(\delta_B, \delta_S, s), s)$ but $\vec{a} \notin \mathcal{CR}(\delta_B \mathbin{\&_{A_u}} \delta_S, s)$.

As an aside, notice that $\vec{a} \in \mathcal{CR}(\delta, s)$ then $\vec{a} \in \mathcal{GR}(\delta, s)$ and for all prefixes $\vec{a'}$ such that $\vec{a'}\vec{b} = \vec{a}$ we have $\vec{a'} \in \mathcal{GR}(\delta, s)$.

Hence, let $\vec{a'} = \vec{a''}a$ such that $\vec{a'} \in \mathcal{GR}(\delta_B \mathbin{\&_{A_u}} \delta_S, s)$ but $\vec{a''}a \notin \mathcal{GR}(\delta_B \mathbin{\&_{A_u}} \delta_S, s)$, and let $\delta''_B = next^*(\delta''_B, \vec{a''}, s)$, $\delta''_S = next^*(\delta_S, \vec{a''}, s)$, and $s' = do(\vec{a''}, s)$.

Since $\vec{a''}a \notin \mathcal{GR}(\delta_B \mathbin{\&_{A_u}} \delta_S, s)$, it must be the case that $next(\delta''_B \mathbin{\&_{A_u}} \delta''_S, a, s'') = \bot$. But then, considering that both $next(\delta''_B, a, s'') \neq \bot$ and $next(\delta''_S, a, s'') \neq \bot$, it must be the case that $\neg A_u(a, s'')$ and exists $\vec{b_u}$ such that $A_u(\vec{b_u}, do(a, s''))$, and $a\vec{b_u} \in \mathcal{GR}(\delta''_B, s'')$ but $a\vec{b_u} \notin \mathcal{GR}(\delta''_S, s'')$.

Notice that $\vec{b_u} \neq \epsilon$, since we have that $a \in \mathcal{GR}(\delta''_S, s'')$. So $\vec{b_u} = \vec{c_u}b_u\vec{d_u}$ with $a\vec{c_u} \in \mathcal{GR}(\delta''_S, s'')$ but $a\vec{c_u}b_u \notin \mathcal{GR}(\delta''_S, s'')$.

Now $\vec{a'} \in \mathcal{GR}(\delta_B \mathbin{\&} mps(\delta_B, \delta_S, s), s)$ and since $A_u(\vec{c_u}b_u, do(\vec{a'}, s))$, we have that $\vec{a'}\vec{c_u}b_u \in \mathcal{GR}(\delta_B \mathbin{\&} mps(\delta_B, \delta_S, s), s)$. Since, $mps(\delta_B, \delta_S, s)$ is controllable for δ_B in s, we have that, if $\vec{a'}\vec{c_u}b_u \in \mathcal{GR}(\delta_B, s)$ then $\vec{a'}\vec{c_u}b_u \in \mathcal{GR}(mps(\delta_B, \delta_S, s), s)$. This, by definition of $mps(\delta_B, \delta_S, s)$, implies $\vec{a'}\vec{c_u}b_u \in \mathcal{GR}(\delta_B \mathbin{\&} \delta_S, s)$, and hence, in turn, $\vec{a'}\vec{c_u}b_u \in \mathcal{GR}(\delta_S, s)$. Hence, we can conclude that $a\vec{c_u}b_u \in \mathcal{GR}(\delta''_S, s'')$, getting a contradiction. □

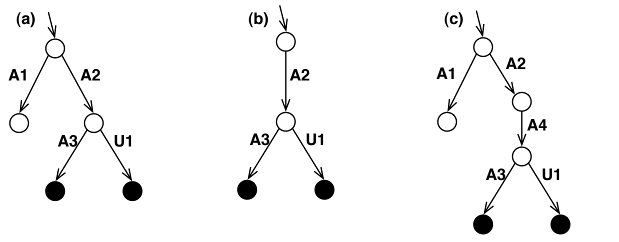


Figure 2: Diagrams of agent behavior specifications δ_{B1} in (a), δ_{B2} in (b), and δ_{B3} in (c).

Examples. Let us illustrate what is involved in obtaining a maximally permissive supervisor in the presence of uncontrollable actions. Suppose that we are in situation S_1 with an agent that has the following behavior (see Figure 2a):

$$\delta_{B1} = A_1 \mid (A_2; (A_3 \mid U_1)),$$

where action U_1 is uncontrollable (i.e. $A_u(a, s) \equiv a = U_1$); we also assume that all actions are always executable. Suppose as well that we have the following supervision specification:

$$\delta_S = (\pi a.(a \neq A_1 \wedge a \neq A_3 \wedge a \neq U_1)?; a)^*; (A_1 \mid A_3),$$

i.e., eventually A_1 or A_3 should be performed, and U_1 should never occur. If we let the agent perform action A_2, we get to situation $do(A_2, S_1)$ with the remaining agent behavior $(A_3 \mid U_1)$ and desired behavior δ_S, where we clearly can no longer effectively control the agent to ensure that it continues to behave as desired. Thus in situation S_1, we have to force the agent to perform A_1. We can in fact do this since A_1 and A_2 are controllable. The maximally permissive controllable supervisor $mps(\delta_{B1}, \delta_S, S_1)$ for this agent in S_1 is simply $\mathbf{set}(\{A_1\})$ (since $\mathcal{CR}(\delta, s) = \{A_1, (A_2; A_3)\}$ and $\mathbf{set}(\{(A_2; A_3)\})$ is not controllable wrt δ_{B1} in S_1).

If instead the agent's behavior in S_1 had been (see Figure 2b):

$$\delta_{B2} = A_2; (A_3 \mid U_1),$$

there would have been no way to ensure that the agent behaved as desired other than ruling out all actions, because even if we can control the agent in S_1, we are not be able to ensure that it continues to behave as desired once it gets to $do(A_2, S_1)$. Indeed $mps(\delta_{B2}, \delta_S, S_1) = \mathbf{set}(\emptyset)$.

The point of the example is that the supervisor must look ahead and always steer the agent away from paths where it cannot be prevented from eventually doing undesirable actions. Our definitions of mps and $\&_{A_u}$ ensure this. Indeed, for our example, we have that $next((\delta_{B1} \mathbin{\&_{A_u}}$

$\delta_S), A_2, S_1) = \perp$ since $next^*(\Sigma(\delta_{B1}), [A_2, U_1], S_1) \neq \perp$ and $next^*(\Sigma(\delta_S), [A_2, U_1], S_1) = \perp$. Thus

$$next((\delta_{B1} \,\&_{A_u}\, \delta_S), a, S_1) \neq \perp \equiv a = A_1.$$

Moreover, $next((\delta_{B1} \,\&_{A_u}\, \delta_S), A_1, S_1) = (nil \,\&_{A_u}\, nil)$ and $Final((nil \,\&_{A_u}\, nil), do(A_1, S_1))$. Thus the only way execute $(\delta_{B1} \,\&_{A_u}\, \delta_S)$ in S_1 is to perform A_1, after which one terminates successfully. For agent behavior δ_{B2} on the other hand, we have

$$\forall a. next((\delta_{B1} \,\&_{A_u}\, \delta_S), a, S_1) = \perp,$$

i.e., all we can do is block.

Note also that in general, one must do lookahead search over the program to find complete executions using $\&_{A_u}$. Consider the following variant of the above example (see Figure 2c):

$$\delta_{B3} = A_1 \mid (A_2; A_4; (A_3 \mid U_1)).$$

In this case, $next((\delta_{B3} \,\&_{A_u}\, \delta_S), A_2, S_1) = ((A_4; (A_3 \mid U_1) \,\&_{A_u}\, \delta_S)$; the resulting program is not final in $do(A_2, S_1)$, yet $next(((A_4; (A_3 \mid U_1) \,\&_{A_u}\, \delta_S), a, do(A_2, S_1)) = \perp$ for all a. However if we do looakead search, we get that

$$next(\Sigma(\delta_{B3} \,\&_{A_u}\, \delta_S), a, S_1) \neq \perp \equiv a = A_1,$$

as well as $next(\Sigma(\delta_{B3} \,\&_{A_u}\, \delta_S), A_1, S_1) = \Sigma(nil \,\&_{A_u}\, nil)$ and $Final(\Sigma(nil \,\&_{A_u}\, nil), do(A_1, S_1))$.

5. CONCLUSION

We have investigated agent supervision in Situation-Determined ConGolog, or SDConGolog, programs. Our account of maximally permissive supervisor builds on the well-established Wonham and Ramadge framework for supervisory control of discrete event systems. However, virtually all work on this framework deals with finite state automata [2, 18], while we handle infinite state systems in the context of the rich agent setting provided by the situation calculus and ConGolog. We used ConGolog as a representative of an unbounded-states process specification language, and it should be possible to adapt our account of supervision to other related languages. We considered a form of supervision that focuses on complete runs, i.e., runs that lead to *Final* configurations. We can ensure that an agent finds such executions by having it do lookahead/search. Also of interest is the case in which agents act boldly without necessarily performing search to get to *Final* configurations. In this case, we need to consider all partial runs, not just good ones. Note that this would actually yield the same result if we engineered the agent behavior such that all of its runs are good runs, i.e. if $\mathcal{RR}(\delta_B, s) = \mathcal{GR}(\delta_B, s)$, i.e., all configurations are final. In fact, one could define a closure construct $cl(\delta)$ that would make all configurations of δ final. Using this, one can apply our specification of the maximally permissive supervisor to this case as well if we replace $\delta_B \,\&\, \delta_S$ by $cl(\delta_B \,\&\, \delta_S)$ in the definition. Observe also, that under the assumption that $\mathcal{RR}(\delta_B, s) = \mathcal{GR}(\delta_B, s)$, in $next(\delta_B \,\&_{A_u}\, \delta_S, a, s)$ we no longer need to do the search $\Sigma(\delta_B)$ and $\Sigma(\delta_S)$ and can directly use δ_B and δ_S.

We conclude by mentioning that if the object domain is finite, then ConGolog programs assume only a finite number of possible configurations. In this case, we can take advantage of the finite state machinery developed for discrete event systems on the basis of [19] (generalizing it to deal with situation-dependent sets of controllable actions), and the recent work on translating ConGolog into finite state machines and back [7], to obtain a program that actually characterizes the maximally permissive supervisor. In this way, we can completely avoid doing search during execution. We leave an exploration of this notable case for future work.

Acknowledgments. We thank Murray Wonham for inspiring discussions on supremal controllable languages in finite state discrete event control, which actually made us look into agent supervision from a different and very fruitful point of view. We also thank the anonymous referees for their comments. We acknowledge the support of EU Project FP7-ICT ACSI (257593).

6. REFERENCES

[1] P. Bertoli, M. Pistore, and P. Traverso. Automated composition of web services via planning in asynchronous domains. *Artif. Intell.*, 174(3-4):316–361, 2010.

[2] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer, second edition, 2008.

[3] G. De Giacomo, Y. Lespérance, and H. J. Levesque. ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121(1–2):109–169, 2000.

[4] G. De Giacomo, Y. Lespérance, and A. R. Pearce. Situation calculus based programs for representing and reasoning about game structures. In *KR*, 2010.

[5] G. De Giacomo, R. Reiter, and M. Soutchanski. Execution monitoring of high-level robot programs. In *KR*, pages 453–465, 1998.

[6] R. Demolombe and E. Hamon. What does it mean that an agent is performing a typical procedure? a formal definition in the situation calculus. In *AAMAS*, pages 905–911, 2002.

[7] C. Fritz, J. A. Baier, and S. A. McIlraith. ConGolog, Sin Trans: Compiling ConGolog into basic action theories for planning and beyond. In *KR*, pages 600–610, 2008.

[8] C. Fritz and Y. Gil. Towards the integration of programming by demonstration and programming by instruction using Golog. In *PAIR*, 2010.

[9] C. Fritz and S. McIlraith. Decision-theoretic Golog with qualitative preferences. In *KR*, pages 153–163, 2006.

[10] A. Goultiaeva and Y. Lespérance. Incremental plan recognition in an agent programming framework. In *PAIR*, 2007.

[11] N. Lin, U. Kuter, and E. Sirin. Web service composition with user preferences. In *ESWC*, pages 629–643, 2008.

[12] S. McIlraith and T. Son. Adapting Golog for composition of semantic web services. In *KR*, pages 482–493, 2002.

[13] P. Ramadge and W. Wonham. Supervisory control of a class of discrete event processes. In A. Bensoussan and J. Lions, editors, *Analysis and Optimization of Systems*, volume 63 of *Lecture Notes in Control and Information Sciences*, pages 475–498. Springer Berlin / Heidelberg, 1984.

[14] R. Reiter. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press, 2001.

[15] S. Sardiña and G. De Giacomo. Composition of ConGolog programs. In *IJCAI*, pages 904–910, 2009.

[16] S. Sohrabi, N. Prokoshyna, and S. A. McIlraith. Web service composition via the customization of Golog programs with user preferences. In *Conceptual Modeling: Foundations and Applications*, pages 319–334. Springer, 2009.

[17] J. Su. Special issue on semantic web services: Composition and analysis. *IEEE Data Eng. Bull.*, 31(3), 2008.

[18] W. Wonham. *Supervisory Control of Discrete-Event Systems*. University of Toronto, 2011 edition, 2011.

[19] W. Wonham and P. Ramadge. On the supremal controllable sub-language of a given language. *SIAM J Contr Optim*, 25(3):637–659, 1987.

Generalized and Bounded Policy Iteration for Finitely-Nested Interactive POMDPs: Scaling Up

Ekhlas Sonu
THINC Lab, Dept. of Computer Science
University of Georgia
Athens, GA. 30602
esonu@uga.edu

Prashant Doshi
THINC Lab, Dept. of Computer Science
University of Georgia
Athens, GA. 30602
pdoshi@cs.uga.edu

ABSTRACT

Policy iteration algorithms for partially observable Markov decision processes (POMDP) offer the benefits of quick convergence and the ability to operate directly on the solution, which usually takes the form of a finite state controller. However, the controller tends to grow quickly in size across iterations due to which its evaluation and improvement become costly. Bounded policy iteration provides a way of keeping the controller size fixed while improving it monotonically until convergence, although it is susceptible to getting trapped in local optima. Despite these limitations, policy iteration algorithms are viable alternatives to value iteration.

In this paper, we generalize the bounded policy iteration technique to problems involving multiple agents. Specifically, we show how we may perform policy iteration in settings formalized by the interactive POMDP framework. Although policy iteration has been extended to decentralized POMDPs, the context there is strictly cooperative. Its generalization here makes it useful in non-cooperative settings as well. As interactive POMDPs involve modeling others, we ascribe nested controllers to predict others' actions, with the benefit that the controllers compactly represent the model space. We evaluate our approach on multiple problem domains, and demonstrate its properties and scalability.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; I.2.8 [**Problem Solving, Control Methods, and Search**]: Dynamic Programming

General Terms

Theory, Performance

Keywords

policy iteration, decision making, multiagent settings

1. INTRODUCTION

Decision making in sequential and partially observable, single-agent settings is typically formalized by partially observable Markov decision processes (POMDP) [11, 19]. In the multiagent context, the decision making is exponentially harder, and depending on the type and perspective of the interaction, is formalized by one of multiple frameworks. In cooperative settings requiring solutions for all agents, decentralized POMDPs [2] sufficiently model the joint decision-making problem. Additionally, interactive POMDPs [7] formalize the decision-making problem of an individual agent in a multiagent setting, which need not be cooperative. Both these frameworks generalize POMDPs in different ways and have relied on extending approximation techniques for POM DPs to their own formalizations for tractability.

One such technique involves searching the solution space directly. Initially proposed in the context of POMDPs [9], the technique represents the solution, called the policy, as a *finite state controller* and iteratively improves it until convergence. The benefit is that the controller typically converges before its value converges across all states and it is useful for an infinite horizon. However, nodes in the controller grow quickly making it computationally difficult to evaluate the controller and continually improve it. Bounded policy iteration (BPI) avoids this growth by keeping the size of the controller fixed as it seeks to monotonically improve the controller's value by replacing a node and its edges with another one [15]. Expectedly, this scales POMDP solutions to larger problems, but the controllers often converge to a local optima. Nevertheless, the benefits of this approach are substantial enough that it has been extended to decentralized POMDPs [3] leading to improved scalability.

In this paper, we introduce a generalization of BPI to the context of finitely-nested interactive POMDPs (I-POMDP) thereby improving on previous approximation techniques on two important fronts: we may solve larger problem domains and generate solutions of much better quality. In contrast to decentralized POMDPs, I-POMDPs do not assume common knowledge of initial beliefs of agents or common rewards, due to which others' beliefs, capabilities and preferences are modeled. They allow for others modeling other agents, and terminate the nesting at some finite level. Recent applications of I-POMDPs testify to its significance and growing appeal. They are being used to explore strategies for countering money laundering by terrorists [12, 13] and enhanced to include trust levels for facilitating defense simulations [17, 18]. They have been used to produce winning strategies for playing the lemonade stand game [21] and even modified to include empirical models for simulating human behavioral data pertaining to strategic thought and action [6].

Being a generalization of POMDPs, solutions of I-POMDPs are also affected by the curses of dimensionality and history

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

that affect POMDPs [14]. The dimensionality hurdle is further aggravated because an agent maintains belief not only over the physical state but also over the models of the other agents, which grow over time as the agents act and observe. Previous approximations for finitely-nested I-POMDPs include interactive particle filtering [4] and interactive point-based value iteration [5]. Particle filtering seeks to mitigate the adverse effect of the curse of dimensionality by forming a sampled, recursive representation of the agent's nested belief, which is then propagated over time. However, its efficiency is still impacted by the number of models because this increases the need for more samples, and it is better suited for solving I-POMDPs with a given prior belief. Interactive point-based value iteration generalizes point-based value iteration [14] to multiagent settings, and reduces the effect of the curse of history. While this approach significantly scales I-POMDPs to longer horizons, we must include all reachable models of the other agent in the state space, which grows exponentially over time, thereby making it susceptible to the dimensionality hurdle. We may also group together models that are behaviorally equivalent resulting in a partition of the model space of the other agent into a finite number of equivalence classes [16]. This approach, analogously to using finite state controllers, allows a compact representation of the model space but computing the exact equivalence requires solving the models.

In generalizing BPI, the interactive state space of the subject agent is reformulated to include the physical states and the set of nodes in a controller without loss of generality. For multiple other agents with differing capabilities and preferences, we include multiple controllers, one for each other agent. Each iteration involves evaluating and possibly improving the nested controller of the other agent followed by improvement of the subject agent's controller. In order to account for the dynamically changing state space, we interleave the evaluation and improvement of controllers at different levels. This approach differs from BPI's implementation in decentralized POMDPs where controllers for each agent are improved independently, but a correlation device is introduced for coordination among them. Such a shared source of randomness may not be feasible in non-cooperative settings. Importantly, we observe that convergence of the subject agent's controller is often dependent on the lower-level controllers converging first.

As we mentioned previously, the benefit is that the space of possible models may be compactly represented using the set of nodes in a controller. On the other hand, the presence of controller(s) embedded in the state space makes evaluation and improvement for the subject agent much more expensive than in the context of POMDPs or decentralized POMDPs. We call our approach *interactive BPI* and experimentally evaluate its properties using benchmark problems. In particular, we show that the converged controller for the subject agent generates solutions of good quality in proportionately less time compared to results reported by the previous best I-POMDP approximation. Ultimately, this allows the application of I-POMDPs to scale to more realistic domains with reduced trade off, as we demonstrate by applying the technique to larger problem domains.

2. BACKGROUND

We briefly review the framework of finitely-nested I-POMDPs and outline previous policy iteration in the context of POMDPs.

2.1 Interactive POMDP

A finitely-nested I-POMDP [7] for an agent i with strategy level, l, interacting with another agent j is defined using the tuple:

$$\text{I-POMDP}_{i,l} = \langle IS_{i,l}, A, T_i, \Omega_i, O_i, R_i, OC_i \rangle$$

where:

- $IS_{i,l}$ denotes the set of *interactive states* defined as, $IS_{i,l} = S \times M_{j,l-1}$, where $M_{j,l-1} = \{\Theta_{j,l-1} \cup SM_j\}$, for $l \geq 1$, and $IS_{i,0} = S$, where S is the set of physical states. $\Theta_{j,l-1}$ is the set of computable, intentional models ascribed to agent j: $\theta_{j,l-1} = \langle b_{j,l-1}, \hat{\theta}_{j,l-1} \rangle$, where $b_{j,l-1}$ is agent j's level $l-1$ belief, $b_{j,l-1} \in \triangle(IS_{j,l-1})$, and $\hat{\theta}_{j,l-1} = \langle A, T_j, \Omega_j, O_j, R_j, OC_j \rangle$, is j's frame. Here, j is assumed to be Bayes-rational. For simplicity, we assume that the frame of agent j is known and remains fixed; it need not be the same as that of agent i. SM_j is the set of subintentional models of j. For the sake of simplicity, in this paper we focus on ascribing intentional models only.
- $A = A_i \times A_j$ is the set of joint actions of all agents.

  The remaining parameters – transition function, T_i, observations, Ω_i, observation function, O_i, preference function, R_i, and the optimality criterion, OC_i – have their usual meaning as in POMDPs [11]. Note that the optimality criterion here is the discounted infinite horizon sum.

An agent's belief over its interactive states is a sufficient statistic, fully summarizing the agent's observation history. Beliefs are updated after the agent's action and observation using Bayes rule. Two differences complicate the belief update in multiagent settings. First, since the state of the physical environment depends on the actions performed by both agents the prediction of how it changes has to be made based on the probabilities of various actions of the other agent. Probabilities of other's actions are obtained by solving its models. Second, changes in the models of other agents have to be included in the update. The changes reflect the other's observations and, if they are modeled intentionally, the update of other agent's beliefs. In this case, the agent has to update its beliefs about the other agent based on what it anticipates the other agent observes and how it updates.

Given the extended belief update, solution to an I-POMDP is a *policy*, analogous to that in a POMDP. Using the Bellman equation, each belief state in an I-POMDP has a value which is the maximum payoff the agent can expect starting from that belief state and over the future. Gmytrasiewicz and Doshi [7] provide additional details on I-POMDPs and how they compare with other multiagent frameworks.

2.2 Policy Iteration

Policy iteration provides an alternative to iterating over the value function, by searching directly over the policy space. While the traditional representation of a policy is as a function from beliefs to actions and iterating over these functions is indeed possible [20], a more convenient representation for the purpose of policy iteration is as a *finite state controller*. At any point in the iteration, the controller represents the infinite horizon policy of the agent.

We may define a simple controller for an agent i, as:

$$\pi_i = \langle \mathcal{N}_i, \mathcal{E}_i, \mathcal{L}_i, \mathcal{T}_i \rangle$$

where $\mathcal{N}_i$ is the set of nodes in the controller, $\mathcal{E}_i$ is the set of edge labels which are observations, Ω_i, in a POMDP, $\mathcal{L}_i$ is the mapping from each node to an action, $\mathcal{L}_i : \mathcal{N}_i \rightarrow A_i$, and $\mathcal{T}_i$ is the edge transition (successor) function, $\mathcal{T}_i : \mathcal{N}_i \times A_i \times \Omega_i \rightarrow \mathcal{N}_i$. For convenience, we group together $\mathcal{E}_i$, $\mathcal{L}_i$ and $\mathcal{T}_i$ in $\hat{f}_i$.

Policy iteration algorithms improve the value of the controller by interleaving steps of evaluating the policy with improving it by backing up the linear vectors that make up the value function. We may view each node in the controller as representing the action and value associated with a vector. As the value function is improved, new vectors may be introduced causing additional nodes in the controller, while some nodes may be dropped if their corresponding vectors are dominated at all states by some other vector [9].

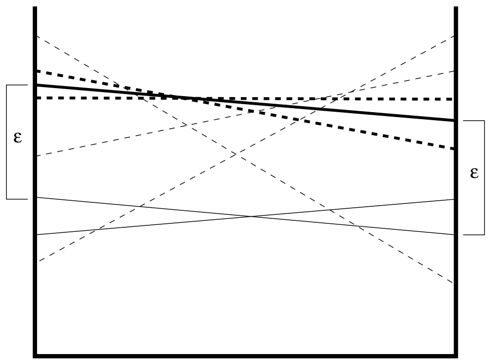


Figure 1: Value vector (solid line in bold), representing a node in the improved controller, is a convex combination of the two dashed backed-up vectors in bold and point-wise dominates a vector that constitutes the value function of the previous controller by ϵ. The dashed vectors constitute the optimal, backed up value function.

Controllers often grow exponentially in size during improvement making evaluation and further improvement intractable. Poupart and Boutilier [15] show that the controller size may be minimized and, in fact kept bounded, in two ways: First, we may replace a node whose corresponding vector is dominated by a convex combination of updated vectors. A convex-combination vector passing through the point of intersection of the combined vectors and parallel to the dominated vector is selected, as we illustrate in Fig. 1. This replaces multiple vectors with a single one and allows us to prune nodes, which previously would not have been removed. This leads to a controller whose transitions due to observations, $\mathcal{T}_i$, may be stochastic. Second, note that if the controller hasn't converged, a backup is guaranteed to improve it. Thus, we may replace some node with another that represents a convex combination of backed up vectors, and whose value is better. This causes the action mapping, $\mathcal{L}_i$, to be stochastic as well. Of course, the technique is susceptible to local optima.

3. GENERALIZED POLICY ITERATION

We generalize the bounded policy iteration technique to the context of I-POMDPs nested to a finite level, l. Notice that a finite state controller partitions the intentional model space, $\Theta_{j,l-1}$, among its nodes. *This is because for a belief in any model in $\Theta_{j,l-1}$, a node exists in the controller that will provide the (boundedly) optimal action(s).* Therefore, the interactive state space, $IS_{i,l} = S \times \Theta_{j,l-1}$, becomes:

$$IS_{i,l} = S \times \mathcal{F}_{j,l-1}$$

where $f_{j,l-1} \in \mathcal{F}_{j,l-1}$ is, $f_{j,l-1} = \langle n_{j,l-1}, \hat{f}_{j,l-1}, \hat{\theta}_{j,l-1} \rangle$. Here, $n_{j,l-1}$ is a node in the set of nodes in the controller, $n_{j,l-1} \in \mathcal{N}_{j,l-1}$; $\hat{f}_{j,l-1}$ is as defined previously in Section 2.2 for a controller; and $\hat{\theta}_{j,l-1}$ is j's frame and is fixed. Notice that the controller represents an initial solution for the entire intentional model space. If there are K other agents with differing capabilities and preferences, the interactive state space becomes, $IS_{i,l} = S \times_{k=1}^{K} \mathcal{F}_{k,l-1}$, where $\mathcal{F}_{k,l-1}$ represents a different controller for each k. This is because the agents differ in their frames and consequently, how their controllers evolve.

Because the set of nodes in $\mathcal{F}_{j,l-1}$ is finite, an important benefit of the above representation is that the infinite model space is represented using a finite node space, thereby making the interactive state space finite as well (assuming that the physical state space is finite). The large model space is often a hurdle for previous approximation techniques that operate on it, such as the interactive point-based value iteration [5]. This motivated arbitrary limitations on the models and on how they evolve, which are no longer necessary. Other, parameterized representations of the model space are also under investigation [8].

Let $\mathcal{F}_{i,l}$ be an initial, level l controller for the subject agent i. Next, we move to evaluating and improving agent i's controller iteratively. Because the controller of the other agent is embedded in i's state space, these steps utilize updated controllers at the lower levels as well thereby generalizing the iterations to multiagent settings.

3.1 Policy Evaluation

As we mentioned previously, each node, $n_{i,l}$, in the controller is associated with a vector of values, $V(\cdot, n_{i,l})$, that gives the expected (converged) value, at each interactive state, of following the controller beginning from that node. In the context of I-POMDPs, this is a $|S \times \mathcal{N}_{j,l-1}|$-dimensional vector for each node. A step of policy evaluation involves computing this vector for each node in the controller. We may do this by solving the following system of linear equations:

$$\begin{aligned} V(s, n_{j,l-1}, n_{i,l}) = & \sum_{a_i \in A_i} Pr(a_i|n_{i,l}) \sum_{a_j \in A_j} Pr(a_j|n_{j,l-1}) \\ & \times \left\{ R_i(s, a_i, a_j) + \gamma \sum_{o_i} \sum_{s'} \sum_{n'_{j,l-1}} T_i(s, a_i, a_j, s')\ O_i(s', a_i, a_j, \right. \\ & o_i) \times \sum_{o_j} O_j(s', a_i, a_j, o_j)\ Pr(n'_{j,l-1}|n_{j,l-1}, a_j, o_j) \\ & \left. \times \sum_{n'_{i,l}} Pr(n'_{i.l}|n_{i,l}, a_i, o_i)\ V(s', n'_{j,l-1}, n'_{i,l}) \right\} \\ & \forall s, n_{j,l-1}, n_{i,l} \end{aligned} \quad (1)$$

In Eq. 1, we compute the expectation over i's actions because multiple actions are possible from a single node of the stochastic controller. Given the multiagent setting, actions of both agents appear in the transition, observation and reward functions in the equation. The terms $Pr(a_i|n_{i,l})$, $Pr(n'_{i.l}|n_{i,l}, a_i, o_i)$ and $Pr(a_j|n_{j,l-1})$, $Pr(n'_{j.l-1}|n_{j,l-1}, a_j, o_j)$ are obtained from $\hat{f}_{i,l}$ and $\hat{f}_{j,l-1}$, respectively, and O_j is obtained from j's frame; all of which are present in $f_{j,l-1}$. Equation 1 is defined for each physical state, s, j's controller node, $n_{j,l-1}$, and i's controller node, $n_{i,l}$. Notice that the update of the other agent's belief is represented using a transition from one node to another by the term, $Pr(n'_{j.l-1}|n_{j,l-1}, a_j, o_j)$.

Solution of the system results in a vector of values for each node in agent i's controller. In the next step, we improve the controller by introducing new nodes with updated value vectors that may uniformly dominate, possibly in combination, those of an existing node and prune the dominated node.

3.2 Policy Improvement

The controller is improved by evaluating whether a node, $n_{i,l}$, in i's controller may be replaced with another whose value, possibly a convex combination of the updated vectors, is better at all interactive states. Instead of first updating the value vectors using the backup operation and then checking for pointwise dominance, Poupart and Boutilier [15] proposed to integrate the two in a single linear program. Our linear program differs from that for a POMDP in involving additional terms related to j's controller. We show this linear program below:

$$
\begin{aligned}
& max \;\; \epsilon \\
& \quad s.t. \; V(s, n_{j,l-1}, n_{i,l}) + \epsilon \leq \sum_{a_i} c_{a_i} \sum_{a_j} Pr(a_j|n_{j,l-1}) \\
& \times \Bigg\{ R_i(s, a_i, a_j) + \gamma \sum_{o_i} \sum_{s'} \sum_{n'_{j,l-1}} T_i(s, a_i, a_j, s') \\
& \times O_i(s', a_i, a_j, o_i) \sum_{o_j} O_j(s', a_i, a_j, o_j) \\
& \times Pr(n'_{j,l-1}|n_{j,l-1}, a_j, o_j) \sum_{n^{o_i}_{i,l}} c_{a_i, n^{o_i}_{i,l}} V(s', n'_{j,l-1}, n^{o_i}_{i,l}) \Bigg\} \\
& \forall \;\; s, n_{j,l-1}; \\
& \sum_{a_i} c_{a_i} = 1; \;\; \sum_{n^{o_i}_{i,l}} c_{a_i, n^{o_i}_{i,l}} = c_{a_i} \;\; \forall a_i, o_i, n^{o_i}_{i,l}; \\
& \;\; c_{a_i, n^{o_i}_{i,l}} \geq 0 \;\; \forall a_i, o_i, n^{o_i}_{i,l}; \; c_{a_i} \geq 0 \;\; \forall a_i
\end{aligned}
\tag{2}
$$

The value function terms in Eq. 2 are obtained from the previous policy evaluation step. We run this linear program for each of i's nodes until a positive ϵ is obtained for a node. $\epsilon > 0$ signals that node, $n_{i,l}$, may be pruned because a convex combination of the backed up value vectors dominate it at least by ϵ at all physical states and nodes of j's controller. Because a single ϵ value is sought for all s, $n_{j,l-1}$, the dominating value vector will be parallel to the pruned one. The solution of the program allows us to construct a new node (say, $n'_{i,l}$) with stochastic actions of agent i as, $Pr(a_i|n'_{i,l}) = c_{a_i}$, and the transition probability to a node ($n^{o_i}_{i,l}$) on performing action a_i and receiving observation o_i as, $Pr(n^{o_i}_{i,l}|n'_{i,l}, a_i, o_i) = c_{a_i, n^{o_i}_{i,l}}$.

We iterate over the evaluation and improvement steps until a positive ϵ is not obtained for any node in i's current controller and the value vectors from Eq. 1 have fixated for every node. Because of the strategy of obtaining a value vector that is parallel to the pruned one, the iterations may converge on a peculiar local optima in which all the value vectors are tangential to the intersections of the exact value function at that step, due to which no further improvement using Eq. 2 is possible. Poupart and Boutilier [15] mention a simple approach of potentially dislodging from the local optima, which is applicable in the context of I-POMDPs as well. Specifically, we pick a belief reachable from the tangential belief and add a node to the controller that corresponds to the value vector associated with the reachable belief. This allows the value of the node representing the tangential vector to improve.

3.3 Nested Controllers

Given that the other agent's controller is embedded in agent i's interactive state space, a naive but efficient approach would be to iteratively improve i's controller while holding j's controller in the state space fixed. However, the corresponding solution will likely be poor as better quality controllers may be available to predict the other agent's actions. This is particularly relevant because I-POMDPs model the other agent as being rational. Subsequently, a more sophisticated approach is to interleave improvements of the other agent's controller with improvements of agent i's controller. However, not only is this approach computationally more intensive, but agent i's interactive state space may change dynamically at every iteration.

Fortunately, the previously detailed approach of BPI keeps the number of nodes fixed as it seeks to improve the controller. Consequently, the size of agent i's interactive state space – and that of j if the level of nesting is greater than 1 leading to embedded controllers in j's interactive state space – remains fixed. Although nodes may be added to j's controller initially or to escape local optima, we perform these iterations before beginning the improvement of the higher-level, agent i's, controller. After this point, the subject agent's interactive state space remains fixed in size, although the individual states may change across iterations due to updates in the stochastic distributions, $\hat{f}_{j,l-1}$.

Finally, at level 0 the I-POMDP collapses into a POMDP. Consequently, we may utilize the traditional BPI [15] for POMDPs in order to evaluate and improve the level 0 agent's controller.

4. ALGORITHM

Algorithm 1 outlines the procedure, labeled Interactive BPI (I-BPI), for performing BPI in the context of finitely nested I-POMDPs. It begins by creating a trivial controller having a single node with a randomly selected action, at each level for agent i or j as appropriate. The interactive state space is then reformulated to include the nodes from the other agent's controller (lines 1-2) as we mentioned previously. In order to apply BPI on a controller of reasonable size, we perform a single full backup at each level to obtain controllers of size $|A_i|$ or $|A_j|$, as appropriate (line 3). If there are more agents with differing frames, then a controller is initialized for each other agent. Subsequent steps are performed for each of these controllers.

Algorithm 2, Evaluate&Improve, then recursively performs

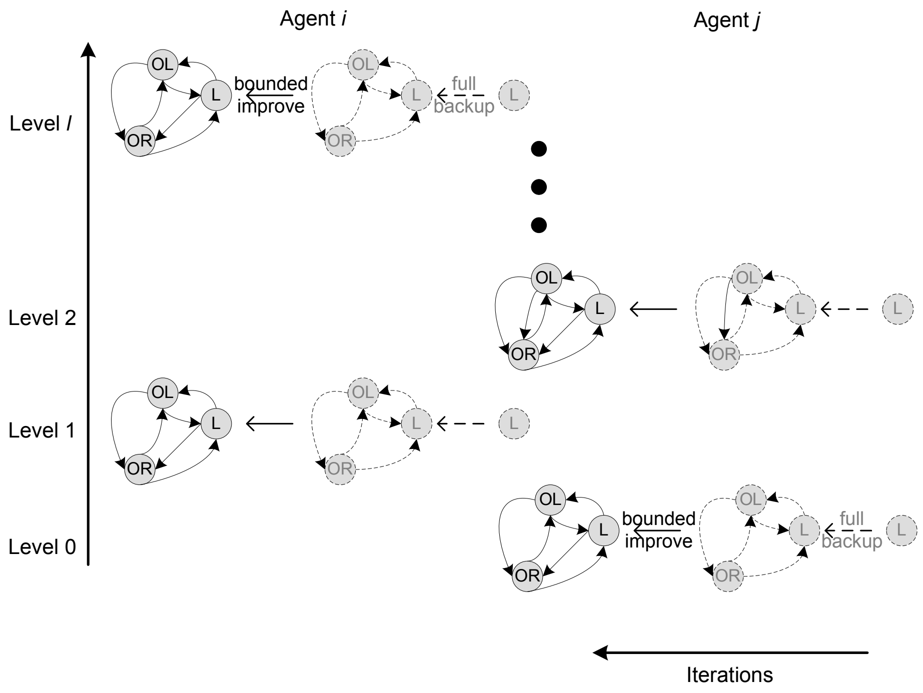


Figure 2: Recursive invocations lead to evaluation and improvement beginning at the bottom and up the nesting levels. Controllers were initialized with a single node and a full backup takes place in the previous iteration (shown dashed and greyed out). Current iteration involves evaluation and bounded improvement of the nested controller bottom up as the recursion unwinds. We demonstrate this process in the context of an example: the well-known multiagent tiger problem. The node labels represent actions of the agents: listen (L), open the left door (OL), and open the right door (OR).

Algorithm 1 Interactive BPI for I-POMDPs

```
Interactive BPI (I-POMDP: $\theta_{i,l}$) returns solution, $\pi^*_{i,l}$

 1: Recursively initialize controllers, $\pi_{i(j),l}$, for both agents,
    such that $|\mathcal{N}_{i(j),l}| = 1$, down to level 0
 2: Reformulate, $IS_{i(j),l} = S \times \mathcal{F}_{j(i),l}$ at each $l$ in $\theta_{i,l}$
 3: Beginning with level, $l = 0$, perform a single-step full
    backup at each level, $l$, resulting in $|\mathcal{N}_{i(j),l}| \leq |A_{i(j)}|$
    nodes in a controller, $\pi_{i(j),l}$
 4: repeat
 5:    repeat
 6:       $\pi_{i,l} \leftarrow$ Evaluate&Improve ($\pi_{i,l}$)
 7:    until no more improvement is possible
 8:    Push controllers at each level from local optima
 9: until no more escapes are possible
10: return converged (nested) controller, $\pi^*_{i,l}$
```

Algorithm 2 Evaluation and bounded improvement of the nested controllers

```
Evaluate&Improve (nested controller: $\pi_{i(j),l}$) returns controller, $\pi'_{i(j),l}$

1: if $l \geq 1$ then
2:    $\pi_{j(i),l-1} \leftarrow$ Evaluate&Improve ($\pi_{j(i),l-1}$)
3: if $l$=0 then
4:    Evaluate controller, $\pi_{i(j),0} = \langle \mathcal{N}_{i(j)}, \mathcal{E}_{i(j)}, \mathcal{L}_{i(j)}, \mathcal{T}_{i(j)} \rangle$
5:    Improve controller, if possible, analogously to a
      POMDP [15]
6: else
7:    Evaluate controller, $\pi_{i(j),l} = \langle \mathcal{N}_{i(j),l}, \mathcal{E}_{i(j),l}, \mathcal{L}_{i(j),l}, \mathcal{T}_{i(j),l} \rangle$, using Eq. 1
8:    Improve controller, if possible, while keeping $|\mathcal{N}_{i(j),l}|$
      fixed using Eq. 2
9: return improved controller, $\pi'_{i(j),l}$
```

a single step of evaluation of the nested controller and its bounded improvement (lines 1-2). For the lowest-level controller, the evaluation and improvement proceeds as outlined by Poupart and Boutilier [15] in the context of POMDPs. At levels 1 and above, we evaluate the controller using Eq. 1 and improve it while keeping the number of nodes fixed using Eq. 2.

The presence of a nested controller leads to novel challenges. Observe that I-BPI interleaves the evaluation and improvement of the controllers at the different levels. The alternate technique would be to evaluate and improve the controller of the lower level until convergence. The former approach better facilitates anytime behavior in comparison to the latter in which the higher-level controller may not be improved for many iterations until the lower-level controller has converged. Of course, the higher-level controllers in the two approaches may not converge to the same local optima. Furthermore, notice that the bounded improvement of j's or i's lower-level controller while keeping the number of nodes fixed still alters the interactive state space because $\hat{f}_{j,l-1}$ or $\hat{f}_{i,l-2}$ changes. Consequently, $IS_{i,l}$ or $IS_{j,l-1}$ may dynamically change at each iteration. Therefore, an alternate technique of evaluating the controllers at all levels first followed by recursively improving them is not feasible because the previous value evaluation of a level l controller is invalidated when lower-level controllers improve.

We illustrate a step within I-BPI on a level l I-POMDP in Fig. 2. On convergence, Algorithm 1 attempts to push the nested controller past any local optima, by escaping it for the lower-level controllers first (line 8). When this is no longer possible, the converged nested controller is returned as the solution of the level l I-POMDP.

Computational Savings In general, the space of models ascribed to the other agent is continuous because each candidate model includes a possible belief as well. I-BPI reformulates the interactive state space by mapping the space of models to a finite set of nodes in the other agent's controller, without loss of generality. However, if we limit the model space and let $|\Theta|$ be a bound on the number of models acribed to one other agent. Then, the interactive state space for K other agents contains $(K|\Theta|)^l$ models at all levels of the nesting. Mapping $|\Theta|$ to $|N|$ nodes of a controller, whose size remains fixed, we obtain a set of size $(K|N|)^l$. This space is significantly smaller because usually, $|N| \ll |\Theta|$, leading to much mitigated impacts of the curses of both dimensionality and history. We empirically demonstrate the effect of these savings next.

5. EXPERIMENTS

We implemented Algorithms 1 and 2 for I-BPI shown in the previous section, and evaluated its properties on two benchmark problem domains: a non-cooperative version of the multiagent tiger problem and a cooperative version of the multiagent machine maintainence (MM) problem, each of which has two agents, i and j. Doshi and Gmytrasiewicz [4] provide details on these problem domains. While these problems have small dimensions, they have been used as benchmarks for previous I-POMDP approximation techniques, such as the interactive particle filter [4] and interactive point-based value iteration (I-PBVI) [5], which employ value iteration. In addition to these toy problems, we evaluate I-BPI's performance and demonstrate scalability using two larger problem domains: autonomous unmanned aerial vehicle (AUAV) reconnaissance problem on a 5×5 grid and the money laundering problem [13], both of which are non-cooperative.

The AUAV problem involves reconnaissance in a 5×5 grid in which an AUAV is tasked with capturing a fugitive who seeks to escape to a safe house (fixed at a predetermined grid location). We model the AUAV as a level 1 I-POMDP. The physical state of the fugitive at any given time is its relative position to the safe house and that of the UAV is its relative position w.r.t. the fugitive. This formulation leads to a physical state space of 81 states for the AUAV and 25 for the fugitive. Each agent may take one of 5 actions of moving in one of the four cardinal directions or listening to get observation about the target's location. We assumed that the actions taken by both agents on the grid are deterministic. The 4 observations for each agent allow it to sense the cardinal direction of its target relative to its own location. We assume that the observations are noisy.

The money laundering problem, introduced by Ng et al. [13], is a game between law enforcement (blue team) and the money launderers (red team) who aim to move their assets from a 'dirty' pot to a 'clean' one through a series of financial transactions while evading capture by the blue team. The blue team can place sensors at various locations such as bank accounts, trusts and real estate to detect the presence of the 'dirty' money. The physical state is defined by the joint location of the dirty money and that of the sensor. The red team may perform any of the three nondeterministic actions of placement, layering or integration to move its assets from one location to another or it could listen to gain noisy information about the location of the blue team's sensor. The blue team may place its sensors in one of eight locations or it could confiscate the assets of the red team. This problem exhibits a physical state space of 99 states for the subject agent (blue team), 9 actions for the subject agent and 4 for the opponent, and 11 observations for the subject and 4 for the other.

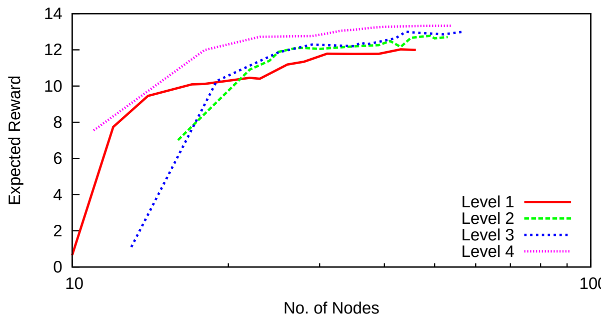


Figure 3: Average rewards for the multiagent tiger problem generally improve until convergence as we allocate more nodes to controllers to facilitate escaping local optima, in I-BPI. As we may expect, controllers generated for more strategically nested I-POMDPs eventually lead to better rewards. Although we do not show the variance for clarity, it tends to be small.

We begin by focusing on the tiger problem, and noting the average reward obtained from *simulating* converged controllers of different node sizes, and of different levels, in Fig. 3. Observe the generally increasing trend of the average rewards as the controllers increase in size on escaping from local optima. This property lends itself to an anytime behavior for I-BPI. Each reward data point is averaged over 5 trials each involving 100 initial beliefs randomly generated, and for each belief, between 100 and 1000 simulation runs were carried out.

Next, in Table 1, we report the average discounted rewards obtained from simulating the controllers that I-BPI generates along with the associated I-BPI run times, as we scale in the context of the number of nesting levels. We compare these rewards with those reported using the previous best I-POMDP approximation technique, I-PBVI [5], where the latter are obtained from actual simulation runs as well. Although I-BPI's controller is for an infinite number of time steps, we limit our runs in the simulations to a finite number of steps in order to compare with I-PBVI. Notice that I-PBVI is able to reasonably scale up to two levels only and the corresponding rewards are significantly lower than those obtained by I-BPI.

As we see from Table 1, I-BPI allows scaling solutions of I-POMDPs up to four levels deep in time duration that is

Problem	Level	Method	Time(s)	Avg. Rwd
Mult. tiger	1	I-BPI	69	11.34
		I-PBVI	2,000	5.34
	2	I-BPI	1,109	12.48
		I-PBVI	696	3.15
	3	I-BPI	3,533	13.00
		I-PBVI	—	—
	4	I-BPI	3,232	13.22
		I-PBVI	—	—
MM	1	I-BPI	15	20.22
		I-PBVI	815	4.86
	2	I-BPI	39	20.55
		I-PBVI	431	3.27
	3	I-BPI	117	21.28
		I-PBVI	—	—
	4	I-BPI	157	21.36
		I-PBVI	—	—
AUAV†	1	I-BPI	7,979	74.08
		I-PBVI	—	—
Money Laun.†	1	I-BPI	1,354	-156.21
		I-PBVI	—	—

Table 1: Average rewards of the controllers at various levels for multiple problem domains. '—' indicates that the corresponding values are not available likely because of scalability issues. †Rewards reported for these larger problem domains are the expected rewards (values) of the corresponding controllers. These results were generated on a RHEL 5 system with Xeon Core2 duo, 2.8GHz each and 4 GB of RAM.

within one hour. It also scales to larger, realistic problem domains. This improvement is primarily due to representing the model space using a finite number of nodes. Comparison with I-PBVI reveals that the quality of the controllers is significantly improved. Furthermore, previous approaches have not scaled solutions beyond two levels.

Next, we demonstrate the performance and scalability of the algorithm on the two large and realistic problems, AUAV reconnaissance and money laundering, in Table 1. Although the latter has been solved previously using the interactive particle filtering [13], the approach assumed an initial belief over the interactive state space and reported run times were more than an order of magnitude greater compared to the time taken by I-BPI. Furthermore, our approach provides a general solution valid over the entire belief space. Table 1 shows the run times for generating converged controllers of good quality for the larger problem domains. The rewards are the expected rewards (values) averaged over 100,000 randomly-generated belief points. While the AUAV problem consumes slightly more than two hours, the money laundering problem takes well within one hour. Converged controllers consisted of 45 nodes for the AUAV problem and 12 nodes for the money laundering problem. Although scaling in the nesting to level two is possible, the corresponding time taken is well beyond our cutoff of two hours. Nevertheless, the reported expected reward is competitive in comparison to those reported by Ng et al. [13] for particular initial beliefs and parameter configurations for money laundering.

An interesting empirical observation in all of these problem domains is that the level l controller, $\pi_{i,l}$, converged – it stops improving and its value vectors obtained by solving the system of linear equations given by Eq. 1 fixate – after the lower-level controller, $\pi_{j,l-1}$, converges.

6. DISCUSSION

As applications emerge for I-POMDPs, approaches that allow its solutions to scale become even more crucial. We introduced a generalized policy iteration algorithm for mutiagent settings in the context of I-POMDPs. This is, to the best of our knowledge, the first policy iteration algorithm proposed for I-POMDPs. We construct a finite state controller for each differing frame of other agents, and models of the other agents get naturally mapped to nodes in the respective controllers. The application of generalized BPI to these controllers ensure that the size of the model space doesn't increase rapidly thereby subduing the effect of the curse of dimensionality, which excessively impacts I-POMDPs.

A limitation of interactive BPI is its convergence to local optima leading to controllers whose quality is unpredictable. While techniques for escaping from local optima may help, this is not guaranteed and the globally optimal value may not be achieved. In particular, the approach of seeking an improved value vector that is uniformly greater than a previous vector leads to multiple local optima; relaxing the constraint of uniform improvement may help.

Another, more practical hurdle was our use of LAPACK++ (`http://math.nist.gov/lapack++`) to solve the system of linear equations in the policy evaluation step. Although LAPACK++ is a popular linear algebra package, it's not optimized for exploiting sparseness in matrices, which we often encountered. We think that further scalability is immediately possible by exploiting the sparsity of the matrices during evaluation. Furthermore, as the number of variables in the linear programs with non-zero values is often low, sparseness may be further exploited in the policy improvement step analogously to the proposal by Hansen [10] in the context of POMDPs. It also seems possible to further improve the performance by accounting for the structure of the controllers. If the controller contains a strongly-connected component, we can evaluate it first thereby focusing on a subset of the nodes, followed by evaluating the rest of the nodes.

Another line of future work is to evaluate the performance of this approach on problem domains having more than two agents. Additionally, the finite state controllers that we use are a type of automata called Moore machines. Recently, Mealy machines were utilized in the context of decentralized POMDPs to good effect [1]. Therefore, another avenue is to investigate the utility of different types of controllers including Mealy machines in our context.

7. ACKNOWLEDGMENTS

This research is supported in part by the NSF CAREER grant, #IIS-0845036, to Prashant Doshi. We thank Kofi Boakye and Brenda Ng for providing the domain files for the money laundering problem.

8. REFERENCES

[1] C. Amato, B. Bonet, and S. Zilberstein. Finite-state controllers based on Mealy machines for centralized and decentralized POMDPs. In *Twenty Fourth AAAI Conference on Artificial Intelligence (AAAI)*, pages 1052–1058, 2010.

[2] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control

of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[3] D. S. Berstein, E. A. Hansen, and S. Zilberstein. Bounded policy iteration for decentralized POMDPs. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1287–1292, 2005.

[4] P. Doshi and P. Gmytrasiewicz. Monte Carlo sampling methods for approximating interactive POMDPs. *Journal of Artificial Intelligence Research*, 34:297–337, 2009.

[5] P. Doshi and D. Perez. Generalized point based value iteration for interactive POMDPs. In *Twenty Third Conference on Artificial Intelligence (AAAI)*, pages 63–68, 2008.

[6] P. Doshi, X. Qu, A. Goodie, and D. Young. Modeling recursive reasoning in humans using empirically informed interactive POMDPs. In *International Autonomous Agents and Multiagent Systems Conference (AAMAS)*, pages 1223–1230, 2010.

[7] P. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multiagent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

[8] Q. Guo and P. Gmytrasiewicz. Modeling bounded rationality of agents during interactions (extended abstract). In *International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, pages 1285–1286, 2011.

[9] E. Hansen. Solving POMDPs by searching in policy space. In *Uncertainty in Artificial Intelligence (UAI)*, pages 211–219, 1998.

[10] E. Hansen. Sparse stochastic finite-state controllers for POMDPs. In *Uncertainty in Artificial Intelligence (UAI)*, pages 256–263, 2008.

[11] L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[12] C. Meissner. A complex game of cat and mouse. *LLNL Science and Technology Review*, pages 18–21, March 2011.

[13] B. Ng, C. Meyers, K. Boakye, and J. Nitao. Towards applying interactive POMDPs to real-world adversary modeling. In *Innovative Applications in Artificial Intelligence (IAAI)*, pages 1814–1820, 2010.

[14] J. Pineau, G. Gordon, and S. Thrun. Anytime point-based value iteration for large POMDPs. *Journal of Artificial Intelligence Research*, 27:335–380, 2006.

[15] P. Poupart and C. Boutilier. Bounded finite state controllers. In *Neural Information Processing Systems*, 2003.

[16] B. Rathnasabapathy, P. Doshi, and P. J. Gmytrasiewicz. Exact solutions to interactive POMDPs using behavioral equivalence. In *Autonomous Agents and Multi-Agent Systems Conference (AAMAS)*, pages 1025–1032, 2006.

[17] R. Seymour and G. Peterson. Responding to sneaky agents in multi-agent domains. In *Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 99–104, 2009.

[18] R. Seymour and G. L. Peterson. A trust-based multiagent system. In *IEEE International Conference on Computational Science and Engineering*, pages 109–116, 2009.

[19] R. Smallwood and E. Sondik. The optimal control of partially observable Markov decision processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.

[20] E. J. Sondik. The optimal control of partially observable Markov processes over the infinite horizon: Discounted cost. *Operations Research*, 26(2):282–304, 1978.

[21] M. Wunder, M. Kaisers, J. Yaros, and M. Littman. Using iterated reasoning to predict opponent strategies. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 593–600, 2011.

Session 3F
Agent-based Software Development

Measuring Plan Coverage and Overlap for Agent Reasoning

John Thangarajah
RMIT University
Melbourne, Australia
john.thangarajah@rmit.edu.au

Sebastian Sardina
RMIT University
Melbourne, Australia
sebastian.sardina@rmit.edu.au

Lin Padgham
RMIT University
Melbourne, Australia
lin.padgham@rmit.edu.au

ABSTRACT

In Belief Desire Intention (BDI) agent systems it is usual for goals to have a number of plans that are possible ways of achieving the goal, applicable in different situations, usually captured by a *context condition*. In Agent Oriented Software Engineering it has been suggested that a designer should be conscious of whether a goal has *complete coverage*, that is, is there some plan that is applicable for every situation. Similarly a designer should be conscious of *overlap*, that is, for a given goal, are there situations where more than one plan could be applicable for achieving that goal. In this paper we further develop these notions in two ways, and then describe how they can be used both in agent reasoning and agent system development. Firstly we replace the boolean value for basic coverage and overlap with numerical measures, and explain how these may be calculated. Secondly we describe a measure that combines these basic measures, with the characteristics of the coverage/overlap in the goal-plan tree below a given goal. We then describe how these domain independent measures can be used for both plan selection and intention selection, as well as for guidance in agent system development.

Categories and Subject Descriptors

I Computing Methodologies [**I.2 Artificial Intelligence**]: I.2.11 Distributed Artificial Intelligence—*Intelligent Agents*

General Terms

Algorithms, Measurement, Design

Keywords

Agent reasoning, intention selection, coverage, overlap, goals, plans

1. INTRODUCTION

In this paper we explore and refine the notions of *coverage* and *overlap* as used in Agent Oriented Software Engineering [8] and describe how they can be used for domain independent agent reasoning. The primary motivation for this work was the search for general purpose characteristics that could sensibly be used to guide intention selection in a BDI (Belief, Desire, Intention) agent system. Typically an agent may well be pursuing multiple goals, and at any point in time it must decide which intention it will progress in the next step. Importantly, the approach we take allows all complex computations to be done at compile time, thus retaining the important soft real-time aspect of agent systems.

Existing agent platforms such as JACK^{TM}[1] offer a choice between *round robin* which does a fixed number of steps on each intention in turn, or *FIFO* which basically uses a queue, finishing one intention and then moving to the next. Some systems such as JAM [6] also allow a priority or utility on goals and/or plans which can be used to order the intention queue. AgentSpeak(XL) [1], AgentSpeak(RT) [13] and TAEMS [5] all use time related information to prioritise the scheduling of intentions, while [15] explores dynamic changes to the priority depending on temporal information. However many standard agent programming languages do not include temporal information - neither when a goal should be achieved by, nor how long a plan or goal can be expected to take. Utility information is also typically unavailable, and requiring a programmer to provide this can be problematic. To our knowledge there is no work which provides any non-temporal general purpose heuristic for intention selection, which does not require user provided priorities or utilities. Agent programming language definitions such as AgentSpeak [9] or CANPLAN [10] typically define only how to step individual intentions and remain agnostic about how to select the particular intention to be progressed.

The intuition behind this piece of work is that (all else being equal) if we have a number of intentions which we are able to progress at any time point, we will prefer to progress the one which we believe has fewest possible successful executions. That is, the one most vulnerable to becoming unable to be successfully executed due to decisions taken in executing other intentions. Preferring such intentions ensures that choices made in pursuing an alternative intention do not eliminate the relatively fewer available ways to achieve such an intention. In order to realise this intuition we use the concept of coverage. In the Agent Oriented Software Engineering context, the Prometheus methodology [8] encourages developers to specify whether a goal has full coverage or not by considering whether for any situation, there can be no applicable plan. In this work we specify coverage as a fraction of the space of all possible models – that portion of the state space for which there is some applicable plan. We calculate this fraction using model counting to ascertain firstly the total number of models in the domain of concern; and secondly the number of models in which a single plan and a set of plans are applicable in using the context conditions of the relevant plans.

As we will discuss in section 3, this *Basic Coverage* must be further refined to give a *Coverage Measure* based on the hierarchy below the goal as well as the immediate relevant plans. As plans are selected, and paths followed, it may be the case that additional

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

[1]http://aosgrp.com/products/jack/

constraints are introduced in sub-plans which in fact means that the applicability space indicated by the context of a plan is reduced further. For example (see figure 2), consider a goal to Travel, with 3 plans: WalkPlan, TramPlan and FlyPlan. Walk has the context condition *distance is short and weather is fine*; Tram has the context condition *distance is short and weather is not fine*; Fly has the context condition *distance is long*. Now assume the TramPlan has a sub-goal GetTimetable which has 3 possible plans: one with context that it is *a weekday and not a public holiday*, one with context that it is *a weekend but not Christmas or Good Friday*, and another that it is *a public holiday but not Christmas or Good Friday*. Here we see that once plans are decomposed there is no decomposition possible for the case of short distance, not-fine weather and Christmas day. Thus the apparent full coverage at the top level goal is compromised. Our Coverage Measure modifies the Basic Coverage to account for such compromises in the tree below.

In section 4, we define a measure of overlap to capture the intuition that in the case where coverage is identical, we are more likely to succeed if a larger part of the state space has alternative plans available in the event that the chosen plan fails (perhaps due to stochastic causes, or maybe because of environmental changes outside of the agent's control).

In section 5, we discuss how these measures can be used for several reasoning tasks, including the intention selection which was the original motivation for the work. The more refined approach to coverage can also enable analysis of agent software to identify and alert the developer to places where the apparent Basic Coverage is substantially reduced once the Coverage Measure which takes into account the hierarchy below is considered. Plan selection is another task which can utilise these measures.

Identification and careful specification of domain independent characteristics can provide the basis for additional power in the next generation of agent systems. Coverage and overlap are important such characteristics and this work provides the foundations for understanding both how to compute them and how they can be used.

2. MEASURING BASIC COVERAGE

We assume the basic standard plan rules typical for agents in the BDI paradigm where $G : \psi \leftarrow P$, meaning that *plan P is a reasonable plan for achieving goal G when (context) condition ψ is believed true*. Though different BDI languages offer different constructs for crafting plans, most allow for sequences of domain actions that are meant to be directly executed in the world (e.g., lifting an aircraft's flaps), and the posting of (intermediate) *sub-goals* $!G'$ (e.g., obtain landing permission) to be resolved. Sub-goals posted during the execution of a plan are resolved via the plan library. When a plan is selected for realising the achievement of a particular goal, it is placed into the *intention base*[2] for execution. The execution of a BDI system can be seen then as a *context sensitive sub-goal expansion*, allowing agents to "act as they go" by making *plan choices* at each level of abstraction with respect to the current situation.

There are then two key decisions an intelligent BDI agent is required to make on an ongoing basis. The first is, which intention from the intention base to progress at each execution cycle. The second is, given a pending goal to be addressed (either completely new or arising from an existing intention), which plan among the available ones in the library to select for execution.

In deciding which plan to select to address a given goal, a BDI system relies on the *context condition* of plans. The context condition ψ of a plan $G : \psi \leftarrow P$ encodes the domain knowledge of when procedure P is an adequate approach to address G. Context conditions are generally formulae in some logical language. For simplicity, at this point, we consider a BDI agent system that is programmed relative to some finite propositional language $\mathcal{P}$.

Given a goal G, let $Pl(G)$ be the set $\{P_1, \ldots, P_n\}$ of alternative plans for achieving G, the so-called *relevant* plans for G. The first thing we are interested in is to know, and compute the coverage of each relevant plan (and even of the goal G itself), that is, in how many world situations an agent will be able to use a plan (or resolve a goal). To make this notion concrete, we recast the coverage problem as that of *model counting* (or #SAT) problem [4], that is, the problem of computing the number of models for a given (propositional) formula – the number of distinct truth assignments to variables for which the formula evaluates to true. As standard, for a propositional formula φ, we will use $\#\varphi$ to denote the model count of φ. The model counting problem generalizes SAT and is the canonical $\#P$-complete problem.

Coverage:
So, given a plan-rule $G : \psi \leftarrow P$, we define

$$\mathcal{A}(P) = \#\psi$$

to be the number of models in which P is applicable (when no ambiguity arises, we use the plan-body P to refer to the whole plan-rule and ψ_P to refer to the corresponding context condition). For example, for plan P_1 in Figure 1, $\mathcal{A}(P)$ denotes the area a+e. When $\mathcal{P}$ is the language in which context conditions are written, we use $S_T = 2^{\mathcal{P}}$ to denote the set of all possible worlds (i.e., models), in the domain of concern. Hence, a single plan as above has a *Basic Coverage*:

$$C(P) = \frac{\mathcal{A}(P)}{S_T}$$

Note that $0 \leq C(.) \leq 1$.

The Basic Coverage can be generalized to a set of plans S in a straightforward manner as follows (recall ψ_X refers to the context-condition of plan X):

$$\mathcal{A}(S) = \#(\bigvee_{P \in S} \psi_P).$$

Considering Figure 1, $\mathcal{A}(\{P_1, P_2, P_3\})$ denotes the areas $a + c + d + e + f + g + h$. As before, the Basic Coverage of a set of plans S is defined as:

$$C(S) = \frac{\mathcal{A}(S)}{S_T}$$

Overlap:
Besides the coverage, we are also interested in overlap, which in terms of Agent Oriented Software Engineering is when there are two or more plans applicable in the same situation to achieve a particular goal. For example, in Figure 1 the region 'e' is the state space covered by both plan P1 and P4. The greater the overlap, the greater the chance of another plan being applicable in the event that one fails, thus providing flexibility and robustness to the whole system. The *Basic Overlap* of a set of plans can be easily defined as:

$$O(\{P_1, \ldots, P_n\}) = \#(\phi_1 \wedge \ldots \wedge \phi_n)$$

It is not difficult to see that all above definitions for coverage and overlap can be computed using model counting solvers. Because the reasoning we are interested in will be done offline and not at BDI execution time, one could make use of exact counting

[2]we describe the structure of the intentions further in section 5.

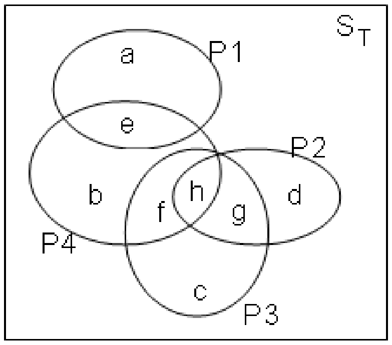


Figure 1: Illustration of state coverage and overlap of 4 alternate plans for achieving a goal.

algorithms, either DPLL-style exhaustive search or those based on "knowledge compilation" [4]. In addition, we observe that, unlike in many SAT/#SAT applications, context conditions in a BDI application are not of a combinatoric nature, and hence we expect exact model counting algorithms to perform well enough for offline preprocessing. Nonetheless, if time is an issue, one could always rely on available approximate counting techniques that provide fast estimates. Some of those techniques come with no guarantees (e.g., ApproxCount [14]), while some methods provide lower or upper bounds with a correctness guarantee, often in a probabilistic or statistical sense (e.g., SampleCount [3]).

We close this section by noting that what we have done so far is to define the coverage and overlap of BDI plans as a model counting problem at the *basic* level. That is, we have not considered the fact that BDI plans can, and most often will, have sub-goals. This calls for a more adequate measure that considers the plans relevant for those sub-goals and that is defined in terms of the whole goal-plan hierarchy implicit in every BDI plan library. This is the subject of the next section.

3. COVERAGE MEASURE OF GOAL-PLAN HIERARCHIES

In this section, we describe the algorithms for calculating a measure of coverage for goals (and plans) considering the goal-plan hierarchies. By goal-plan hierarchy we refer to the implicit goal-plan structure that is present in the plan library of BDI systems – goals are achieved by a set of alternate plans, and each plan contains sub-goals or actions,[3] where each sub-goal in turn is handled by a set of plans. Figure 2 illustrates an example of a goal-plan tree structure based on our motivating example from Section 1. We note we restrict our attention to plan libraries having no cycles.

We want to calculate a *Coverage Measure* for goals which modifies the Basic Coverage defined in the previous section based on the goal-plan hierarchy beneath each plan of the goal. As mentioned previously, an apparently high Basic Coverage may be compromised by lower coverage in the underlying tree. We take this into account when determining the Coverage Measure of a goal as follows (we use the diagram illustrated in Figure 1).

Exclusive Coverage and Exclusive Overlap.

Firstly we define two notions to allow us to speak about the number of models in each of the types of region a–h in Figure 1. We define the notion of *Exclusive Coverage* ($EC(G, P)$) of a plan, P, with respect to a goal, G, to capture the number of models in which P is exclusively applicable (i.e. ψ_P is exclusively true), as defined as in Equation 1. For example, regions 'a', 'b', 'c' and 'd' in Figure 1.

We define *Exclusive Overlap*($EO(G, S)$) in Equation 2 to capture the number of models in the overlapping area of all plans in some group S with respect to a goal G. Note that the exclusive overlap for a group of plans refers to the number of models in the region that is exclusively covered by *all* of the plans in that group only and no other plan. For example, for the group {P2,P3} the exclusive overlap region is 'g' and for the group {P2,P3,P4} it is 'h'.

Recall from the previous section that $\mathcal{A}(P)$ is the number of models in which the plan P is applicable (similarly, $\mathcal{A}(\{P_1, \ldots, P_n\})$ for a set of plans), and that S_T is the total number of models in the domain of concern.

$$EC(G, P)^4 = \frac{\mathcal{A}(Pl(G)) - \mathcal{A}(Pl(G) \setminus \{P\})}{S_T} \tag{1}$$

$$EO(G, S)^5 = \frac{\mathcal{A}(Pl(G))) - \mathcal{A}(Pl(G) \setminus S) - \sum_{P \in S} EC(G, P)}{S_T} \tag{2}$$

Note that if a plan has no overlap with the other plans, then its basic and exclusive coverage measures coincide.

Coverage Measure for Goals and Plans.

We now wish to define a *Coverage Measure* for a goal by summing these separate regions, appropriately discounted with regard to the Coverage Measure of the underlying tree.

The exclusive coverage areas are discounted by the Coverage Measure of the relevant plan (which captures the Coverage Measure of its sub-goals). The exclusive overlap areas (regions 'e','f','g' and 'h') are discounted by the Coverage Measure of the plan with the highest Coverage Measure of that group. This is because when there is an overlap between plans all of them are applicable for that space and the agent would always choose the plan with the highest Coverage Measure (i.e. highest chance of success). The Coverage Measure of a goal G, denoted $C_M(G)$, is therefore the addition of the Coverage Measures of the exclusive coverage regions of each plan of G and the Coverage Measures of the exclusive overlap regions of each grouping of plans, with the appropriate discount factors described above. This is defined as follows:

$$\begin{aligned} C_M(G) = & \\ & \sum_{P \in Pl(G)} EC(G, P) \times C_M(P) \; + \\ & \sum_{S \in \mathcal{P}_{\geq 2} Pl(G)} EO(G, S) \times \max(\{C_M(P) \mid P \in S\}). \end{aligned} \tag{3}$$

The Coverage Measure of a plan P, denoted $C_M(P)$ and shown in Equation 4, is defined as the product of the Coverage Measures of its sub-goals, where $Sg(P)$ is the set of sub-goals within the body of plan P. We take the product since for a plan to succeed all the sub-goals must be achieved, and we assume that the success (with respect to coverage) of each sub-goal is independent of each other's success. Using probability theory, the probability of two independent events occurring together (in parallel or in sequence) is the

[3]Messages to other agents are regarded as actions for the purpose of this paper.

[4]In terms of model counting, as presented in Section 2, $EC(G, P)$ can be expressed as $\#(\psi_P \wedge \bigwedge_{P' \in Pl(G) \setminus \{P\}} \neg\psi_{P'})$.

[5]In terms of model counting $EO(G, S)$ can be expressed as $\#(\bigwedge_{P \in \Pi} \psi_P \wedge \bigwedge_{P' \in Pl(G) \setminus \Pi} \neg\psi_{P'})$.

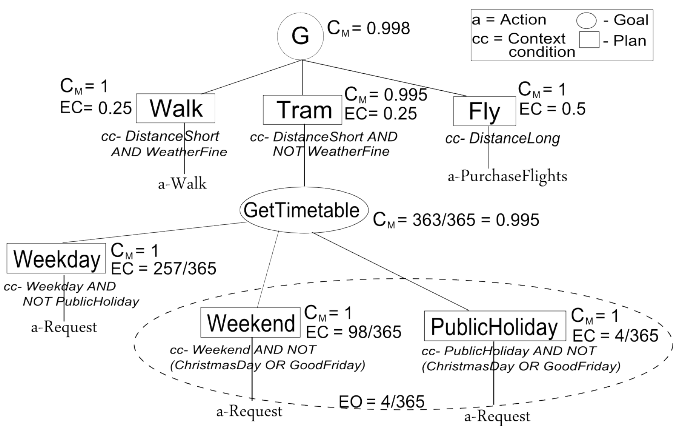


Figure 2: Example coverage calculation for a goal. [7]

product of the individual probabilities of each event. A plan with no sub-goals (that is, a leaf plan in a goal-plan tree with only actions) has a Coverage Measure of 1.

$$C_M(P) = \begin{cases} 1 & \text{if } Sg(P) = \{\} \\ \prod\limits_{G \in Sg(P)} C_M(G) & \text{otherwise} \end{cases} \tag{4}$$

In the above Coverage Measure for a plan, except for the case where the sub-goals have a Coverage Measure of 1, the more sub-goals a plan has the lower the Coverage Measure of that plan. As stated above, this is due to the fact that the more the agent has to achieve the lesser the chance of success. This highlights the importance that as plans are modularized into sub-goals[6], it is important to consider the individual coverage of each of the sub-goals striving to achieve full coverage whenever possible.

Figure 2 illustrates the above calculations applied to the travel goal example introduced in Section 1. As mentioned then, if we only considered the coverage of the immediate plans of the top level goal, the goal would have full coverage ($0.25 + 0.25 + 0.5 = 1$). However, due to the incomplete coverage at the lower levels (there is no 'Tram' plan decomposition for ChristmasDay or GoodFriday) this value is reduced when Coverage Measures are calculated.

4. OVERLAP MEASURE OF GOAL-PLAN HIERARCHIES

We now consider how an *Overlap Measure* should be calculated, using a similar notion of influence from the underlying goal-plan tree, as with our Coverage Measure. We recall that overlap is useful in the case of plan failure, so an alternative plan can be tried in the event that one fails. Hence, overlap can be seen as a measure related to likely success of the goal with respect to failure recovery.

Recall from Section 2 that the overlap of a goal is the overlap between the plans of the goal. That is, the models in which more than one of the plans are applicable in. We recall also that Exclusive Overlap of a set of plans relative to a particular goal ($EO(G, S)$) is defined in Equation 2.

In order now to calculate our Overlap Measure for a goal, taking account of the underlying tree, we will:

1. Firstly, sum the Exclusive Overlap count of the individual regions in all combinations of plans in $Pl(G)$, relative to $\mathcal{A}(Pl(G))$[8], multiplied by the average of the Coverage Measure of the plans involved. We discount using the Coverage Measure, as the overlap is only of value to the extent that there is coverage, and we consider the average Coverage Measure of the group of plans and not the maximum Coverage Measure, since overlap is beneficial in the event of plan failure hence the success of all the plans needs to be considered.

2. Secondly, in order to capture the amount of overlap in the tree below each plan of the goal, we add the sum of the Overlap Measures of the plans of the goal ($Pl(G)$). The Overlap Measure of each plan (Equation 6) is determined by the sum of the Overlap Measures of the sub-goals of the plan (if any).

This is defined as follows:

$$O_M(G) = \tag{5}$$
$$\sum_{S \in \mathcal{P}_{\geq 2} Pl(G)} \frac{EO(G,S)}{S_T} \times AVG\big(\{C_M(P) \mid P \in S\}\big) + \sum_{P \in Pl(G)} O_M(P)$$

Equation 6, defines the Overlap Measure of a plan. Unlike in coverage, where the coverage measure of sub-goals discount the coverage measure of the plan, in overlap we are concerned with the total overlapping spaces in all the sub-goals (that is, the tree beneath the plan) to provide a measure related to success in the event of failure.

$$O_M(P) = \sum_{G \in Sg(P)} O_M(G) \tag{6}$$

We note that whereas C_M is always between 0 and 1, the Overlap Measure is ≥ 0.

In the above Overlap Measure we do not distinguish between the number of plans that overlap a particular region of the state space (for example, in Figure 1 region 'e' is overlapped by 2 plans, and region 'h' is overlapped by 3 plans). Although having more plans overlapping the same space may seem better in terms of recovering from multiple failures, this depends on how much failure is expected and it is not clear that it makes sense to consider such a level of detail. However, if desired or considered important for a particular application, Equation 5 can be modified to weight the Overlap Measure of each exclusive overlap region proportionally to the number of plans in the overlap, by replacing term $EO(G, S)$ with $|S| \times EO(G, S)$.

We note that a property of our definition of the Overlap Measure is that, under the assumption of complete coverage for each goal, it does not matter how the overlap is distributed within the tree; nor is the Overlap Measure affected by the form of the goal-plan tree (i.e. depth or breadth). For example, in Figure 3 we assume that the plans P3 and P4 overlap 30% of the relevant space for G2, but have no sub-goals. In the tree under G1 this same overall amount of overlap is distributed with 10% overlap between P1 and P2, and

[6]This is often the case when developing agent systems.

[7]Note that this is not a complete example and is a simplified version for illustration and clarity of the Coverage Measure calculations. Similarly, we have made assumptions about the number of public holidays in the year and the overlap of public holidays with weekends and weekdays.

[8]Note that we are concerned with a measure relative to the models covered by (the plans of) the goal, not relative to S_T as in coverage.

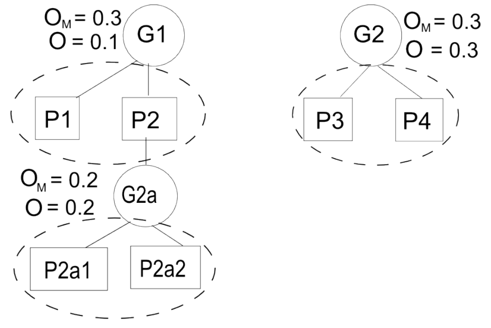


Figure 3: Example of possible distribution of overlap.

20% overlap between P2a1 and P2a2. Applying our formulae[9] we see that $O_M(G1)$ is equal to $O_M(G2)$.

If we assume constant, but less than full coverage then the Overlap Measure will be affected by the shape of the tree, and the distribution of the overlap within that tree, in a similar way to how the Coverage Measure is affected. With trees such as are shown in Figure 3 an overlap of say 40% in the tree of G2 will give a greater Overlap Measure than the same size overlap at G2a, when the Overlap Measure is propagated up to G1.

5. COVERAGE & OVERLAP FOR AGENT REASONING

In this section we discuss the usage of the Coverage and Overlap Measures that we have introduced in the previous section in agent reasoning. In particular, how they may be used for plan selection, intention selection and in agent design and development.

In order to perform some of the above reasoning, the goal-plan structures for each top level goal and the Basic Coverage counts (as defined in Section 2) can be constructed at compile time. (For details on constructing goal-plan trees with annotations see [11].)

5.1 Plan Selection

The most straightforward use of the Coverage and Overlap Measures at execution time would be to select between a set of applicable plans, that is, when there is more than one plan applicable in the current state to achieve a goal. Intuitively, we would prefer to choose the plan with the highest Coverage Measure, and if Coverage Measures were equal, then the one with a higher Overlap Measure. We capture this below.

Let $App(G)$ be set of applicable plans of G ($App(G) \subseteq Pl(G)$) and $Pref(P)$ be the preference of plan P. The *Pref* partial ordering of plans within an applicable plan set is then defined by the following rules.

1. $\forall_{P, P' \in App(G)}\ C_M(P) > C_M(P') \implies Pref(P) > Pref(P')$;

2. $\forall_{P, P' \in App(G)}\ C_M(P) = C_M(P') \wedge O_M(P) > O_M(P') \implies Pref(P) > Pref(P')$.

These can readily be incorporated into a plan selection rule of an agent language such as that of CAN [10].

[9]We note that the Coverage Measure of all the plans will always be 1, because leaf plans are 1, and under the assumption of full coverage at each goal, there is no discounting as one goes up the tree.

5.2 Intention Selection

Intention selection is the issue of, if an agent has a number of intentions (instantiated plan structures for achieving high level goals) that are active, how should these be interleaved, and in particular which one should be progressed at the next step. Programming languages such as AgentSpeak [9] typically define when an intention is in a state that it can be progressed, along with how to progress it, but do not define how to select between multiple intentions. As one intention is progressed and its goals realised, it is of course possible that things are changed in such a way that other intentions are unable to be successfully realised.

Current implementations of BDI agents typically use one of several defaults in progressing intentions. One method ("FIFO") is to simply place intentions in a queue, and execute each in turn - though moving to the next if one becomes idle for some reason. Another approach is what is known as "round robin" where each intention in the queue is progressed a fixed number of steps before moving onto the next. An additional option is to (somehow) assign a priority or utility to intentions and order the queue according to that "priority". However, using priorities or utilities typically requires substantial information to be provided by the developer, which is generally onerous for anything more than a simple priority on high level goals. To our knowledge there is no principled mechanism for determining this priority, that does not rely on either temporal information, or programmer provided utilities.

We explain below how our Coverage Measure can be used to select which intention to work on next, when the current intention either finishes or becomes *unprogressable*.

When a goal is adopted as an intention, an instance of its goal-plan tree is created and placed into the set of intentions (Γ) that the agent wishes to accomplish. We refer to this instantiated goal-plan tree as the *execution-tree* (*ExTree*) of that goal. As an intention (that is, the adopted goal) is progressed[10] the nodes of its execution-tree are annotated as follows:

- When a plan is selected.
- When a plan completes. That is, when all its sub-goals and actions complete.
- When a goal completes. That is, when at least one of its plans completes.
- When a plan fails. That is, for an abstract plan when one of its actions fail, and for a concrete plan when one of its sub-goals fail.[11]

Figure 4 shows an example of a goal-plan tree (a) and a corresponding execution-tree (b) that is partially executed with the above annotations.

We define the function *next*(I), where $I = ExTree(G)$, to return the next step in the execution-tree of the adopted goal G. For example, in Figure 4(b) the *next*(I) function would return the action a_4.

The intention structure Γ is then a set of execution-trees of the corresponding top level goals, *each execution-tree representing an intention*. An intention is said to be *progressable* if the next step of the execution is either an action or a sub-goal for which there is at least one applicable plan in the current state.

[10]or executed.

[11]In the interest of space we do not go into a definition of the various reasons a goal could fail. For a fully usable language this requires language constructs which are outside the scope of this paper, such as tests on beliefs, which in turn require updates of beliefs, etc.

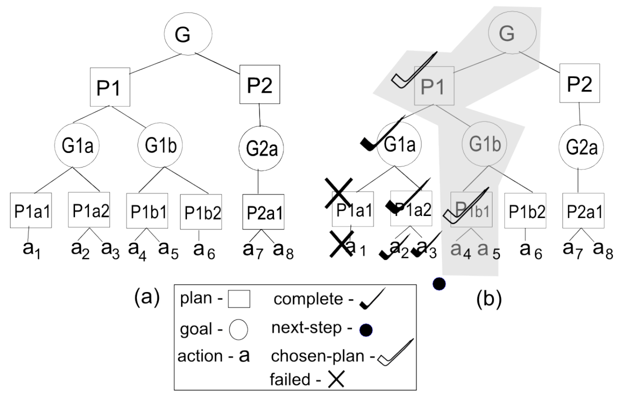


Figure 4: Illustration of goal-plan tree and execution tree.

When an intention I is *progressed* this involves processing the *next*(I) in the execution-tree. If that item is an action, it is simply executed. If it is a goal, then a plan is selected from the applicable plan set for that goal and the *next*(I) becomes the first sub-goal or action of that plan. It is here that when more than one plan is applicable the coverage based plan selection mechanism described earlier in this section may be used.

An intention is said to be *finished* when the root goal of the execution-tree completes as described above, or fails.[12] An intention is said to be *unprogressable* if it is in a waiting state. This can be due to a wait for a message response, or some action executing externally that needs to complete before proceeding. It may also include such things as an intention being blocked [10] or suspended for some reason [12].

Once an agent has selected an intention for execution we assume it will continue to be progressed until it is either finished, or becomes unprogressable. When this happens, a new intention must be chosen to start executing. It is here that we will use our Coverage Measures.

The intention selection question is then, when the current intention finishes or becomes unprogressable, which progressable intention should be made current - or which progressable intention has the highest priority. Our intuition is that we will prefer to prioritise things with least coverage, as these are the ones that have fewer possibilities for success. The Coverage Measure for an intention that has not yet started to execute is simply the Coverage Measure of the goal, relevant to that intention, as described in Section 3. However, we also need a Coverage Measure for a partially executed intention, that is an execution-tree that has been progressed, which we obtain by building on our previous definitions as follows:

1. The Coverage Measure of a goal, if a plan has been selected, is the Basic Coverage of the chosen plan ($C(P)$), multiplied by the Coverage Measure of that plan ($C_M(P)$).

$$C_M(G) = \Big(C(P) \bullet C_M(P)\Big) \qquad \text{if } \exists P : \text{chosen-plan}(G, P) \quad (7)$$

2. The Coverage Measure of a (possibly partially executed) plan is the product of the Coverage Measures of the goals still to be achieved of that plan.

$$C_M(P) = \begin{cases} 1 & \text{if } Sg(P) = \{\} \\ \prod\limits_{g \in Sg(P) \wedge \text{not complete(g)}} C_M(g) & \text{otherwise.} \end{cases} \quad (8)$$

This Coverage Measure for partially executed intentions (Equation 7) captures our strong preference, once we have made a plan selection, to succeed that plan without failure or backtracking. Thus we no longer consider the coverage of alternative plans in measuring the coverage of a goal, where a plan selection has been made. It also captures the fact (in Equation 8) that coverage of sub-goals which have already succeeded is irrelevant. For example, in Figure 4(b), the shaded region indicates the part of the goal-plan which is considered when calculating the Coverage Measure of the partially executed intention.

The following rules now provide a priority ordering on progressable intentions which can be applied as needed, but in particular for choosing a new current intention when an intention finishes or becomes unprogressable.

Recall Γ is the set of intentions (execution-trees) and that the next step of an intention (progressed in the corresponding execution-tree) is either an action or sub-goal. So, let

- $Act(next(I))$ be true when the next step of intention I is an action;
- $Sg(next(I))$ be true when the next step of intention I is a sub-goal;
- $Current(I)$ be true when I is the current intention being progressed;
- $G(I)$ be the top level goal of the intention I, and $Pr(I)$ be the priority of intention I.

The relative priority ordering of intentions are established by the following rules applied in the oder specified below:

1. $\forall_{I, I' \in \Gamma}\, Current(I) \implies Pr(I) > Pr(I')$.
2. $\forall_{I, I' \in \Gamma}\, Act(next(I)) \wedge Sg(next(I')) \implies Pr(I) > Pr(I')$.
3. $\forall_{I, I' \in \Gamma}\, C_M(G(I)) > C_M(G(I')) \implies Pr(I) > Pr(I')$.
4. $\forall_{I, I' \in \Gamma}\, O_M(G(I)) > O_M(G(I')) \implies Pr(I) > Pr(I')$.

Our first priority is then to maintain focus as long as is possible while the second is to execute any actions that are pending, as it makes no sense to keep decomposing plans without executing the actions in as timely manner as possible. The third priority then captures the intuition that if one has an intention that has relatively fewer spaces/models in which it can be progressed through to completion, then we prefer to progress it when we have the opportunity, as opposed to an intention which has a higher Coverage Measure, representing a larger number of models incorporating successful completion. Finally, if other things are equal, we prioritise doing first the one that has a smaller Overlap Measure - i.e. the one with fewer options for recovery.

5.3 Agent Design and Development

When specifying events (goals) within an agent design, using the Prometheus methodology [8] and the supporting Prometheus Design Tool (PDT) [7] developers specify (amongst other things) the goal-plan trees for each agent. During this process they are

[12]As with goal failure, due to space limitations and the focus of this paper, we do not go into a precise definition of failure of an intention.

prompted to consider the coverage and overlap properties of the goal. It is suggested developers note if there is overlap, on what basis one of the overlapping plans will be chosen, and if there is not full coverage, which are the situations where there will be no applicable plan. This is because both lack of full coverage, and unintended overlap are common causes of bugs in implementations of BDI agents systems. In this section we identify areas in which the the coverage and Overlap Measures we have defined may be beneficial.

Coverage Measures to identify potential flaws in the design: With the new measures defined in this paper it is now possible to alert developers to cases where lack of coverage elsewhere in the goal-plan tree compromises full (or a partial high level) coverage at a top level goal. Areas where there is a significant difference between the Basic Coverage of a goal, and the Coverage Measure would be candidates for further investigation.

Overlap as a measure of robustness of goals: Overlap, as discussed, is related to the potential to recover from failure during execution, selecting an alternative plan to try to achieve some failed sub-goal. Once overlap figures are obtained, at compile time, it becomes possible to report which goals have a relatively low Basic Overlap and/or Overlap Measure. Alternative overlapping plans are one way of making a system robust to stochastic failure. While an important high level goal may have a high Basic Overlap to support such failure recovery, a low Overlap Measure may indicate potential for increasing robustness in the tree below.

Overlap for debugging: Overlap measures can also be useful in potential debugging. As noted previously, the reason for prompting developers to consider overlap is partly because unintended overlap is a common cause of error. If a system is failing at some top level goal, a non-zero Overlap Measure may indicate that the tree below that goal is a potential place to examine for error.

5.3.1 Abstract Coverage Measures

As mentioned above during the *detailed design* phase of developing agent systems, the developer specifies the goal-plan tree and for each goal indicate whether full coverage and overlap is expected. However, currently these attributes are not used for any automated reasoning during design, although they are used for testing and debugging. The coverage and overlap measures as we have defined them in this paper require at least a precise specification of context conditions, along with the domain and range of all variables used, which is not necessarily available at design time.

However, the Coverage Measure that we have defined can be abstracted to give some useful information at design time, using only boolean values (`True, False` as currently provided by the developer) for initial measures, and three values (`True, False, Uncertain`) for calculated measures. These measures can then indicate places where full Basic Coverage is compromised by the tree below and can call the designer's attention to possible areas for further examination prior to any implementation. We describe this abstraction below.

In order to specify the rules for this more Abstract Coverage Measure ($C^A_M(.)$) we first define an ordering over the three values we will use:

`True > Uncertain > False`

The rules are then as follows:

1. The Abstract Basic Coverage of a goal is `True`, for full coverage or `False` otherwise.

$$C^A(G) = \begin{cases} \texttt{True} & \text{if there is full coverage indicated} \\ \texttt{False} & \text{otherwise} \end{cases} \quad (9)$$

2. The Abstract Coverage Measure of an plan is the minimum of the Coverage Measure of its sub-goals or `True` if the plan has no sub-goals.

$$C^A_M(P) = \begin{cases} \texttt{True} & \text{if } Sg(P) = \{\} \\ \text{MIN}\left\{ C^A_M(g) \mid g \in Sg(P_A) \right\} & \text{otherwise} \end{cases} \quad (10)$$

3. The Coverage Measure of a goal is as follows (here, `T=True`, `F=False`, and `U=Uncertain`):

$$C^A_M(G) = \begin{cases} \texttt{T} & \text{if } \forall P \in Pl(G)\ C^A_M(P) = \texttt{T} \wedge C^A(G) = \texttt{T} \\ \texttt{F} & \text{if } \forall P \in Pl(G)\ C^A_M(P) = \texttt{F} \vee C^A(G) = \texttt{F} \\ \texttt{U} & \text{otherwise} \end{cases} \quad (11)$$

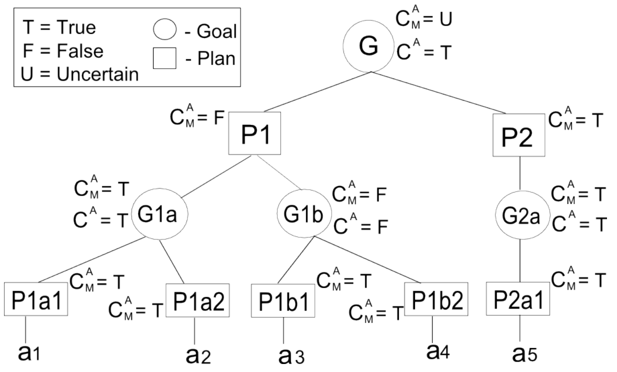


Figure 5: Example Abstract Coverage Measures for design

Figure 5 illustrates the propagation of these simplified Coverage Measures in a goal-plan tree. The `Uncertain` Coverage Measure value thus identifies the case where there is some path(s) with full coverage through the entire tree, but ensuring full plan coverage is dependent on plan selections. For example, in Figure 5 the path containing plan P2 has full coverage.

6. CONCLUSION

BDI Agent programming languages provide a powerful platform for developing complex applications. They support the use of domain specific information, which makes them very suitable for real and complex applications. However, their value also lies in the generic reasoning that is incorporated into the execution engine, independently from the domain based program. The key standard features on which much of the success of the paradigm is based, are hierarchical plan selection based on context conditions and persistent goals with failure recovery. Additional generic mechanism that can be incorporated into the execution engine to make these systems smarter, without requiring application specific coding, are of interest to the agent development community. In trying to identify such opportunities it is also important to be cognisant of the need to avoid overloading the developer with requirements to provide details which are not readily available.

In this work we have taken the concepts of coverage and overlap that have been used in Agent Oriented Software Engineering and refined these to support smarter agent systems. Importantly we do not require any additional information from the developer or the domain, beyond what is required in typical BDI agent development. Also importantly, all the complex calculation can be done offline at compile time, leaving only simple computational update processes during execution.

The basis of our approach is the process to calculate, using model counting, a numerical measure of the extent to which the set of plans for a goal cover the relevant state space. We recognise however that apparently high coverage at the immediate level can be compromised by lack of coverage in the sub-goals below. Consequently we define a measure which takes account of this factor, and discounts the immediate coverage based on the characteristics of the underlying goal-plan tree. We apply a similar approach to measuring overlap.

Having defined these measures we then show how they can be used for both plan selection and intention selection at execution time. In addition, we indicate how the measures can be used to identify potential Software Engineering issues. Based on the quantitative approach developed we then abstract back to a qualitative approach suitable for use during design, prior to full details being available to calculate numerical Coverage Measures. This provides better information than what is currently provided in agent design methodologies.

In this work we use an idealised and simplified agent programming language, and in particular we do not account for any decisions coded within plan bodies. Consequently it is possible that actual coverage is less than what is calculated with our method. However, we do not consider this a substantial disadvantage, as it is possible to replace test and action steps with new sub-goals whose plans have the test condition and action precondition, respectively, as context conditions. This would allow the coverage measure to detect the otherwise hidden constraints. Under certain assumptions, also, constraints in plan bodies could be automatically regressed to plans' context conditions (e.g., see [2]), though this is an orthogonal problem and is out of the scope of this work.

We also acknowledge that we have not yet implemented the reasoning described, based on these measures, and so do not yet have experience of their value in practice. Nevertheless, based on many years practical experience, and work with industry partners, we are convinced that these measures provide valuable information which, either alone, or in combination with additional aspects, can further improve the behaviour of autonomous intelligent agents.

Acknowledgments

We acknowledge the support of the Australian Research Council under Discovery grant DP1094627.

7. REFERENCES

[1] R. H. Bordini, A. L. C. Bazzan, R. de Oliveira Jannone, D. M. Basso, R. M. Vicari, and V. R. Lesser. AgentSpeak(XL): Efficient intention selection in BDI agents via decision-theoretic task scheduling. In *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1294–1302. ACM Press, 2002.

[2] L. P. de Silva, S. Sardina, and L. Padgham. First principles planning in BDI systems. In C. Sierra, C. Castelfranchi, K. S. Decker, and J. S. Sichman, editors, *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS)*, volume 2, pages 1001–1008, Budapest, Hungary, May 2009. IFAAMAS.

[3] C. P. Gomes, J. Hoffmann, A. Sabharwal, and B. Selman. From sampling to model counting. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2293–2299, 2007.

[4] C. P. Gomes, A. Sabharwal, and B. Selman. Model counting. In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 633–654. IOS Press, 2009.

[5] B. Horling, V. Lesser, R. Vincent, and T. Wagner. The Soft Real-Time Agent Control Architecture. *Autonomous Agents and Multi-Agent Systems*, 12(1):35–92, 2006.

[6] M. J. Huber. JAM: A BDI-theoretic mobile agent architecture. In *Proceedings of the Annual Conference on Autonomous Agents (AGENTS)*, pages 236–243, New York, NY, USA, 1999. ACM Press.

[7] L. Padgham, J. Thangarajah, and M. Winikoff. Prometheus design tool. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1882–1883, 2008.

[8] L. Padgham and M. Winikoff. *Developing Intelligent Agent Systems: A Practical Guide*. Wiley Series in Agent Technology. John Wiley and Sons, NY, USA, 2004.

[9] A. S. Rao. Agentspeak(L): BDI agents speak out in a logical computable language. In W. V. Velde and J. W. Perram, editors, *Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World. (Agents Breaking Away)*, volume 1038 of *Lecture Notes in Computer Science (LNCS)*, pages 42–55. Springer, 1996.

[10] S. Sardina and L. Padgham. A BDI agent programming language with failure recovery, declarative goals, and planning. *Autonomous Agents and Multi-Agent Systems*, 23(1):18–70, 2011.

[11] J. Thangarajah. *Managing the Concurrent Execution of Goals in Intelligent Agents*. PhD thesis, RMIT University, Melbourne, Australia, 2004.

[12] J. Thangarajah, J. Harland, D. Moreley, and N. Yorke-Smith. Suspending and resuming tasks in BDI agents. In *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 405–412, Estoril, Portugal, May 2008.

[13] K. Vikhorev, N. Alechina, and B. Logan. Agent programming with priorities and deadlines. In *Proceedings of the Tenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 397–404, 2011.

[14] W. Wei and B. Selman. A new approach to model counting. In *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing (SAT)*, volume 3569 of *Lecture Notes in Computer Science*, pages 324–339. Springer, 2005.

[15] H. Zhang, S. Y. Huang, and Y. Chang. An agent's activities are controlled by his priorities. In *Proceedings of the 2nd KES International conference on Agent and multi-agent systems: technologies and applications*, KES-AMSTA'08, pages 723–732, Berlin, Heidelberg, 2008. Springer-Verlag.

Programming Norm-Aware Agents

Natasha Alechina
School of Computer Science
University of Nottingham
Nottingham NG8 1BB, UK
nza@cs.nott.ac.uk

Mehdi Dastani
Department of Information and Computing Sciences
Universiteit Utrecht
3584CH Utrecht, The Netherlands
M.M.Dastani@uu.nl

Brian Logan
School of Computer Science
University of Nottingham
Nottingham NG8 1BB, UK
bsl@cs.nott.ac.uk

ABSTRACT

Normative organisations provide a means to coordinate the activities of individual agents in multiagent settings. The coordination is realized at run time by creating obligations and prohibitions (norms) for individual agents. If an agent cannot meet an obligation or violates a prohibition, the organisation imposes a sanction on the agent. In this paper, we consider *norm-aware* agents that deliberate on their goals, norms and sanctions before deciding which plan to select and execute. A norm-aware agent is able to violate norms (accepting the resulting sanctions) if it is in the agent's overall interests to do so, e.g., if meeting an obligation would result in an important goal of the agent becoming unachievable. Programming norm-aware agents in conventional BDI-based agent programming languages is difficult, as they lack support for deliberating about goals, norms, sanctions and deadlines. We present the norm-aware agent programming language N-2APL. N-2APL is based on 2APL and provides support for beliefs, goals, plans, norms, sanctions and deadlines. We give the syntax and semantics of N-2APL, and show that N-2APL agents are rational in the sense of committing to a set of plans that will achieve the agent's most important goals and obligations by their deadlines while respecting its most important prohibitions.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Programming Languages and Software

General Terms

Languages, Theory

Keywords

Agent programming languages, Normative systems

1. INTRODUCTION

Normative organisations, e.g., [5], provide a means to coordinate the activities of individual agents in a multiagent system. In a normative organisation, coordination is realised at run time by creating obligations and prohibitions (norms) for individual agents. An obligation requires an agent to bring about a particular state of the environment by a specified deadline, while a prohibition requires the agent to avoid bringing about a particular state before a deadline. If an agent cannot meet an obligation or violates a prohibition, the organisation imposes a sanction on the agent.

In general, norms imposed on an agent by a normative organisation may conflict with the agent's existing goals. In such a situation, a rational agent must choose between its existing goals and the norms imposed by the organisation. We say an agent is *norm-aware* if it can deliberate on its goals, norms and sanctions before deciding which plan to select and execute. A norm-aware agent is able to violate norms (accepting the resulting sanctions) if it is in the agent's overall interests to do so, e.g., if meeting an obligation would result in an important goal of the agent becoming unachievable. As an example, consider an agent that has agreed to review papers for a conference. The normative system in this case is the conference organisation, and the sanction for not discharging reviewing obligations by the review deadline may be reputational damage (e.g., being put on a blacklist). Let us further assume that the reputational damage for being late with reviews for an important conference such as AAMAS is greater than that incurred for being late with reviews for informal workshops. A norm-aware rational agent may still consider defaulting on its obligations to review for AAMAS if it acquires a more important goal with a tighter deadline, such as attending to some family emergency. Note that while we assume the severity of sanctions (and hence the priority or importance of obligations and goals) can be compared, their values are not necessarily commensurable in the sense that we cannot say whether being late with AAMAS reviews incurs the same sanction as being late with reviews for, e.g., two informal workshops.

There has recently been considerable work on programming frameworks for developing normative organisations [5, 14, 9]. Such frameworks are often designed to inter-operate with existing BDI-based agent programming languages, e.g., [2, 4]. However, programming norm-aware agents in conventional BDI-based agent programming languages remains difficult, as such languages typically lack support for deliberating about goals, norms, sanctions and deadlines.

In this paper we present a BDI-based agent programming language N-2APL for norm-aware agents. N-2APL extends 2APL [4] with support for normative concepts including obligations, prohibitions, sanctions, deadlines and durations. We give the syntax and operational semantics of N-2APL and explain how it supports norm-aware deliberation. We show that agents programmed in N-2APL are *norm-aware rational*, and that key assumptions underlying the design of N-2APL are necessary in the sense that if they are relaxed, a N-2APL agent is either no longer norm-aware rational, or the agent's deliberation about goals, norms and sanctions is intractable.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

2. NORMATIVE SYSTEMS

We conceive a multiagent system as consisting of a set of agents interacting with each other and with a shared environment. In normative multiagent systems, the interaction between agents and the environment is governed by a normative exogenous organisation, which is specified by a set of conditional norms with their associated deadlines and sanctions. Individual agents decide which action to perform in the environment after which the state of the environment changes. Subsequently, the organisation evaluates the environmental changes with respect to the conditional norms to: determine any obligations to be fulfilled or prohibitions that should not be violated by the agents (termed detached obligations and prohibitions); determine any violations based on the deadlines of the detached obligations and prohibitions; and impose any corresponding sanctions. The role of the exogenous organisation is thus to continuously 1) monitor the behaviour of agents, 2) evaluate the effects they bring about in the environment, 3) determine norms that should be respected, 4) check if any norm is violated, and 5) take necessary measures (i.e., imposing sanctions) when norms are violated. This continuous process is often implemented by a so-called control cycle [5].

A conditional obligation is expressed as a tuple

$$\langle c, O(\iota, o), d, s\rangle$$

with the intuitive reading "if condition c holds in the current state of the environment then there is an obligation for agent ι to establish an environment state satisfying o before deadline d, otherwise agent ι will be sanctioned by updating the environment with s". In the conference reviewing example, a possible norm is to return reviews for a paper assigned to a reviewer. An instance of such a conditional norm could be represented as:

$$\langle \mathit{collect}\ \&\ p_{id}Ar_i, O(r_i, p_{id}Rev), \mathit{notify}, BLr_i\rangle$$

which indicates that when a conference is in the review collection phase (*collect*) and paper p_{id} is assigned to reviewer r_i ($p_{id}Ar_i$), then the reviewer r_i is obliged to return the review of paper p_{id} ($p_{id}Rev$) before the notification phase (*notify*) starts. Violating this norm results in the reviewer being put on a blacklist (BLr_i), damaging its reputation.

A conditional prohibition is expressed as a tuple

$$\langle c, F(\iota, p), d, s\rangle$$

with the intuitive reading "if condition c holds in the current state of the environment, then it is forbidden for agent ι to establish an environment state satisfying p before deadline d, otherwise sanction s will be imposed." Unlike obligations, where a sanction is incurred once if the obligation is not discharged by the deadline, in the case of prohibitions, the agent incurs a sanction each time the prohibition is violated. For example, if it prohibited to submit a paper longer than 8 pages, the agent will incur a sanction (e.g., the paper being rejected without review) each time it submits a paper longer than 8 pages. In what follows, we consider the simpler case of prohibitions without deadlines (i.e., prohibitions with an indefinite deadline or with $\perp$ as deadline).

As our focus is on an agent's decision problem when operating in a normative multiagent system, we ignore the working of the normative exogenous organisation, and simply assume an additional step in the control cycle through which the organisation broadcasts detached obligations and prohibitions in the form of events (i.e., a set of normative events) to each agent to whom the norms are directed. A detached obligation event broadcast to agent ι has the form:

$$\mathit{obligation}(\iota, o, d, s)$$

with the intuitive reading "agent ι is obliged to establish an environment state satisfying o before deadline d, otherwise it will be sanctioned by updating the environment with s". For example, when a paper is assigned to a reviewer during the collection phase, the organisation generates and sends the following (singleton) set of detached obligations to the corresponding reviewer r_i:

$$\{\mathit{obligation}(r_i, p_{id}Rev, \mathit{notify}, BLr_i)\}$$

We assume that deadlines associated with detached obligations can be mapped to real time values expressed in some appropriate units that specify the time by which the obligation should be discharged. For example, *notify* might map to the real time value "5pm on Friday" which specifies the time by which the review should be returned. Similarly, a detached prohibition event broadcast to agent ι has the form:

$$\mathit{prohibition}(\iota, p, s)$$

with the intuitive reading "the agent ι is prohibited from establishing an environment state satisfying p, otherwise it will be sanctioned by updating the environment with s"

Of course, obligations or prohibitions from a normative organisation may conflict with an agent's goals or with other obligations or prohibitions it has already received (possibly from another normative organisation). For example, if paper reviewing and family emergencies cannot be attended to concurrently, the agent needs to schedule its intentions in some order, such as: first deal with the emergency, then review paper 1, then review paper 2, etc. Some schedules will be better than others. For example, if paper 1 has an earlier review deadline than paper 2, and the agent still has sufficient time after dealing with the emergency to review paper 1, and (after reviewing paper 1) it still has time to review paper 2, the schedule above may be optimal. On the other hand, if the agent does not have enough time left to review both papers by their deadlines and reviewing paper 2 is a more important obligation (incurs a greater sanction), then rationally it should review paper 2 rather than paper 1 next.

We assume that an agent has a preference or priority ordering over goals and sanctions that determines whether it is more important (from the point of view of the agent) to, e.g., achieve a goal g or to avoid the sanction s associated with an obligation or prohibition. In the case of goals, the priority indicates the importance of *achieving* the goal state, while in the case of sanctions, the priority indicates the importance of *avoiding* the sanction state, i.e., sanctions that entail a smaller penalty for the agent will have lower priority. Sanctions thus induce an order on obligations and prohibitions: the priority of an obligation or prohibition is determined by the priority of the sanction that would be incurred if the obligation is not discharged by its deadline or the prohibition violated.[1] We assume that an agent's preferences over goals and sanctions are ordered on an ordinal scale, i.e., that it is always preferable to achieve a higher priority goal (or avoid a high priority sanction) than to achieve any number of lower priority goals (or avoid any number of lower priority sanctions). (As we show in section 5, if this is not the case, the agent's deliberation about norms is intractable.)

If the agent's goals and obligations are not jointly achievable or its goals cannot be achieved without violating one or more prohibitions, the agent uses the relative priority of goals and sanctions to determine which goal(s) to drop or which sanction(s) to incur. Following BOID [3], we will consider special cases of the priority ordering and define them as specific agent types. For example, a

[1]Note that we do not assume that obligations are preferred or desirable states for agent; rather the agent is motivated by the avoidance of sanctions rather than the achievement of obligations.

social agent can be characterized by a priority ordering that prefers all obligations to its goals and a selfish agent can be characterized by a priority ordering that prefers its goals over its obligations.

If the agent's goals and obligations are not jointly achievable, a norm-aware rational agent should schedule its intentions so as to achieve goal and obligation states with highest priority. Programming norm-aware rational agents is non-trivial, as it depends on the deadlines of the goals and obligations, the plans the agent has to achieve its goals and obligations, whether its plans violate any prohibitions, the expected execution time of the agent's plans and the extent to which the plans can be executed in parallel.

3. THE LANGUAGE N-2APL

In this section we present N-2APL, an agent programming language for norm-aware rational agents. N-2APL is a modification of the agent programming language 2APL [4] which supports normative concepts including obligations, prohibitions, sanctions, deadlines and durations. We first briefly present 2APL, focusing on those elements modified in N-2APL, before describing the extended programming constructs of N-2APL and how it supports norm-aware deliberation.

3.1 2APL: a Brief Summary

2APL is a BDI-based agent programming language that allows the implementation of agents in terms of cognitive concepts such as beliefs, goals and plans. A 2APL agent program specifies an agent's initial beliefs, goals, plans, and the reasoning rules it uses to select plans (`PG-rules`), to respond to messages and events (`PC-rules`), and to repair plans whose executions have failed (`PR-rules`). The initial beliefs of an agent includes the agent's information about itself and its surrounding environment. The initial goals of an agent consists of formulas each of which denotes a situation the agent wants to realize (not necessarily all at once). The initial plans of an agent consists of tasks that an agent should initially perform.

In order to achieve its goals, an 2APL agent adopts plans. A plan consists of basic actions composed by sequence, conditional choice, conditional iteration and non interleaving operators. The non interleaving operator, $[\pi]$ where π is a plan, indicates that π is an *atomic* plan, i.e., the execution of π should not be interleaved with the execution of any other plan. Basic actions include external actions (which change the state of the agent's environment); belief update and goal adopt actions (which change the agent's beliefs and goals), and abstract actions (which provide an abstraction mechanism similar to procedures in imperative programming).

Planning goal rules allow an agent to select an appropriate plan given its goals and beliefs. A planning goal rule $\langle pgrule \rangle$ consists of three parts: the head of the rule, the condition of the rule, and the body of the rule. The body of the rule is a plan that is generated when the head (a goal query) and the condition (a belief query) of the rule are entailed by the agent's goals and beliefs, respectively. Procedure call rules are used to select a plan for an abstract action and to handle external events. Plan repair rules are used to revise plans whose executions have failed.

3.2 N-2APL Extensions and Restrictions

To support norm-aware agents, N-2APL extends some key constructs of 2APL and restricts or changes the semantics of others. We briefly describe these changes below. The syntax of N-2APL is shown in Figure 1 in EBNF notation. Programming constructs in bold are exactly the same as in 2APL. Due to lack of space, we omit the detailed specification of these programming constructs, which can be found in [4].

Beliefs in N-2APL are exactly the same as in 2APL and consist of Horn clause expressions. Goals in 2APL may be conjunctions of positive literals. In N-2APL we restrict goals to single atoms and extend their syntax to include optional deadlines. A *deadline* is a real time value expressed in some appropriate units which specifies the time by which a goal should be achieved. We write a goal g with a deadline d as $g : d$. If no deadline is specified for a goal as part of the agent's program, we assume a deadline of infinity.

In N-2APL, non-atomic plans are the same as in 2APL and consist of basic actions composed by sequence, conditional choice, and conditional iteration operators. However in N-2APL we change the interpretation of the non interleaving operator: $[\pi]$ indicates that the execution of π should not be interleaved with the execution of other *atomic* plans (rather than not interleaved with the execution of *any* other plan as in 2APL). In N-2APL, atomic plans are assumed to contain basic actions that may interfere only with the basic actions in other atomic plans, e.g., belief update actions that update the same belief(s), or external actions that change the position of the agent etc. For example, for a particular agent, reviewing a paper may require the agent's undivided attention and cannot be executed in parallel with reviewing another paper. However other plans, such as having lunch and taking a train can be executed in parallel (the agent can have a sandwich on the train while reviewing the paper). As illustrated by the example, in N-2APL, we also allow external actions in different non-atomic plans to be executed in parallel, rather than interleaved as in 2APL (see section 4 for details). Lastly, we restrict the scope of the non interleaving operator such that non-atomic plans cannot contain atomic sub-plans, either directly or through the expansion of an abstract action, i.e., plans to achieve top-level goals are either wholly atomic or non-atomic. As we show in section 5, these changes are necessary for the agent's deliberation about norms to be tractable.

We extend the syntax of plans in the body of a PG rule to include an optional field specifying the time required to execute the plan proposed by the PG rule. In N-2APL, a PG rule has the form:

$$\kappa \leftarrow \beta \mid \pi : t$$

where κ is a goal query, β is a belief query, π is a plan and t is the time required to execute π. We assume that the time required to execute a plan is primarily determined by the time required to execute the external actions it contains, i.e., that the amount of time required to execute internal actions (belief update, goal adopt, abstract actions etc.) is small compared to the time required to execute external actions. The problem of determining t for a plan π therefore reduces to the problem of determining the sequence of external actions that will be executed, and estimating the time required to execute each of these external actions. For simplicity, we assume that the time required to execute each plan π is fixed and known in advance.

As explained in section 2, the creation of an obligation or prohibition by an external organisation causes an event to be sent to the agent. An obligation event, represented as $obligation(\iota, o, d, s)$, specifies the time d by which the obligation o must be discharged, i.e., its deadline, and the sanction, s, that will be applied if the obligation is not discharged by the deadline. A prohibition event, represented as $prohibition(\iota, p, s)$, specifies a prohibition p that must not be violated and the sanction s that will be applied if execution of the agent's plans violates the prohibition. Obligations and prohibitions are added to the agent's goal and event bases, respectively. In particular, an obligation is adopted as a goal $o : d$ with priority corresponding to (the importance of avoiding) s, and a prohibition is represented by a prohibition event $prohibition(p, s)$ where the priority of p corresponds to (the importance of avoiding)

```
⟨Agent_Prog⟩    =  [ "Beliefs:" { ⟨belief⟩ } ],
                   [ "Goals:" ⟨goals⟩ ],
                   [ "Plans:" ⟨plans⟩ ],
                   [ "PG-rules:" { ⟨pgrule⟩ } ],
                   [ "PC-rules:" { ⟨pcrule⟩ } ]
                   [ "PR-rules:" { ⟨prrule⟩ } ]
⟨goals⟩         =  ⟨goal⟩ { ","⟨goal⟩ } ;
⟨goal⟩          =  ⟨atom⟩ ":" ⟨deadline⟩;
⟨pgrule⟩        =  ⟨goalquery⟩ "<-" ⟨belquery⟩ "|" ⟨plan⟩ ":" ⟨duration⟩;
⟨goalquery⟩     =  ⟨goalquery⟩ "and" ⟨goalquery⟩ | ⟨goalquery⟩ "or" ⟨goalquery⟩ | "(" ⟨goalquery⟩ ")" | ⟨atom⟩ ;
⟨belquery⟩      =  ⟨belquery⟩ "and" ⟨belquery⟩ | ⟨belquery⟩ "or" ⟨belquery⟩ | "(" ⟨belquery⟩ ")" | ⟨literal⟩ ;
⟨plan⟩          =  ⟨atomic-plan⟩ | ⟨non-atomic-plan⟩;
⟨atomic-plan⟩   =  "["⟨non-atomic-plan⟩"]";
⟨sanction⟩      =  ⟨atom⟩;
⟨deadline⟩      =  ⟨time⟩;
⟨duration⟩      =  ⟨int⟩;
```

Figure 1: EBNF syntax of N-2APL.

the sanction s (see section 4). We assume the programmer provides a binary relation $\mathit{pref}(x, y)$ where x, y may be goals, obligations or prohibitions that returns true if the goal (sanction) x has higher priority than the goal (sanction) y, and that the order induced by pref is stable. Finally, we assume that it is possible to define a function $\mathit{effects}(\pi)$ which returns the set of literals appearing in the postconditions of all external actions in π. A plan π violates a prohibition p iff $p \in \mathit{effects}(\pi)$, i.e., if executing the plan would cause p to become true.

3.3 Norm-aware Deliberation

In determining which plan to adopt for a goal, a norm-aware agent must take into account (and possibly revise) plans to which it is currently committed. In addition it must decide when each plan to which it is committed should be executed, i.e., it must schedule the execution of its plans. Informally, a schedule is an assignment of a start or next execution time to a set of plans which ensures that: all plans complete by their deadlines, at most one atomic plan executes at any given time, and where the goals achieved and the prohibitions avoided are of the highest priority.[2]

To define a schedule, we first define a set of feasible plans. A set of plans $\Pi = \{\pi_1, \ldots, \pi_n\}$ is *feasible* iff:

1. it is possible for each plan to complete execution before its deadline, that is, for each plan $\pi_i \in \Pi$

$$ne(\pi_i) + et(\pi_i) - ex(\pi_i) \leq dl(\pi_i)$$

   where $ne(\pi_i)$ is the time at which π_i will next execute, $et(\pi_i)$ is the time required to execute π_i, $ex(\pi_i)$ is the time π_i has spent executing up to this point and $dl(\pi_i)$ is the deadline for π_i (we assume that all plans π_i complete by $et(\pi_i)$ and hence $et(\pi_i) - ex(\pi_i)$ is always non-negative);

2. if π_i is an atomic plan, then no other atomic plan is scheduled to execute concurrently with π_i, namely the set $\{\pi_j \in \Pi \setminus \{\pi_i\} \mid (ne(\pi_i) < ne(\pi_j) + et(\pi_j)) \wedge (ne(\pi_j) < ne(\pi_i) + et(\pi_i))\}$ contains no atomic plans; and

3. if π_i is a currently executing atomic plan whose deadline has not passed, for any plan $\pi_j \in \Pi \setminus \{\pi_i\}$, $ne(\pi_j) > now + et(\pi_i) - ex(\pi_i)$, i.e., an atomic plan cannot preempt a currently executing atomic plan.

[2]In what follows, in the interests of brevity we shall often refer to the 'deadline' and 'priority' of a plan π rather than the deadline of the goal achieved by π or the priority of the goal achieved or the sanction avoided by executing π.

Feasibility determines which sets of plans can be executed by their deadlines without violating atomicity constraints.

The agent commits to a feasible set of plans that is preference-maximal. If it is not possible to execute all plans by their deadlines it will drop plans that achieve goals of lower priority in preference to plans which achieve goals of higher priority. Similarly, it will drop plans that violate prohibitions of higher priority than the goal achieved by the plan.

To make this precise, we define a *preference-maximal* set of plans as follows. Consider a set of plans $\Pi = \{\pi_1, \ldots, \pi_n\}$ and prohibitions $\Delta = \{p_1, \ldots, p_m\}$. $\Gamma \subseteq \Pi$ is preference-maximal (with respect to Π and Δ) iff:

1. Γ is feasible;
2. $\forall \pi_i \in \Pi$ such that $\pi_i \notin \Gamma$, either
   - $\{\pi_i\}$ is infeasible, or
   - $\exists \Gamma' \subseteq \Gamma$: the minimal priority of a plan in Γ' is greater than or equal to the priority of π_i, and $\{\pi_i\} \cup \Gamma'$ is infeasible; or
   - $\exists p_j \in \Delta, p_j \in \mathit{effects}(\pi_i)$ and the priority of p_j is greater than or equal to the priority of π_i

Intuitively, this definition describes a subset of Π which is 'maximally feasible' (no more plans from Π can be added if the plans are to remain feasible) and moreover, plans in $\Pi \setminus \Gamma$ cannot be scheduled together with some subset of Γ that contains plans(s) of higher priority or they violate a prohibition in Δ of the same or higher priority.

A *schedule* is a preference-maximal set of plans together with their start (next execution) times.

The agent uses a *schedulability criterion* when deliberating about which plan to adopt for a goal, and which plans to drop. In order to decide if a PG-rule

$$\kappa \leftarrow \beta \mid \pi : t$$

is applicable, we check that: $\gamma \models_g \kappa$ (i.e., that $g\,\tau \models \kappa$ for some $g : d \in \gamma$ and substitution τ), $\sigma \models \beta$ and $\exists \Gamma \subseteq \Pi$ such that $\Gamma \cup \{\pi\}$ is preference-maximal relative to Π, Δ, where γ is the agent's goal base, σ is the agent's belief base, Π is the agent's plan base and Δ is the agent's prohibitions. The first two conditions are straightforward and simply check that the agent has a plan which will achieve the goal and that plan is applicable in the current belief context. The third condition is more complex: it checks whether adopting the plan π is rational for the agent. Adopting π is rational if π together with some subset Γ of the agent's plan base is feasible, and any plans $\pi' \in \Pi \backslash \Gamma$ that must be dropped in order to schedule π by its deadline have strictly lower priority than the goal or obligation achieved by π.

A N-2APL agent thus adopts an open-minded commitment strategy — at any given point in its execution its plan base comprises a preference-maximal set of plans. Observe that if an agent is 'social' and orders its obligations before any of its own goals, then all obligations will be discharged by the agent provided it has feasible plans to achieve them. On the other hand, selfish agents which prioritize their own goals, may incur (in the worst case, when they do not attend to any of the obligations and violate all prohibitions) all the sanctions possible in the normative system.

3.4 Scheduling Algorithm

Algorithm 1 Scheduling Algorithm

```
 1: function SCHEDULE($\Pi, \Delta$)
 2:     $\Gamma_s := \emptyset, \Gamma_p := \emptyset$,
 3:     for all $\pi \in \Pi$ in descending order of priority do
 4:         $V := effects(\pi) \cap \Delta$
 5:         if $\neg atomic(\pi)$ then
 6:             if $now + rt(\pi) \leq dl(\pi) \wedge$
 7:                 $pr(\pi) \geq \operatorname{argmax} pr(p),\ p \in V$ then
 8:                 $ne(\pi) := now$
 9:                 $\Gamma_p := \Gamma_p \cup \{\pi\}$
10:             end if
11:         else
12:             if $executing(\pi)$ then
13:                 $ne(\pi) := now$
14:                 $\Gamma'_s := \Gamma_s$
15:             else
16:                 $t := now$
17:                 $\Gamma'_s := \emptyset$
18:                 for all $\pi' \in \Gamma_s$ do
19:                     if $dl(\pi') \leq dl(\pi)$ then
20:                         $\Gamma'_s := \Gamma'_s \cup \{\pi'\}$
21:                         $t := \max(ne(\pi') + rt(\pi'), t)$
22:                     else
23:                         $ne(\pi') := ne(\pi') + rt(\pi)$
24:                         $\Gamma'_s := \Gamma'_s \cup \{\pi'\}$
25:                     end if
26:                 end for
27:                 $ne(\pi) := t$
28:             end if
29:             if $\forall \pi_i \in \Gamma'_s \cup \{\pi\}$: $now + \sum_{ne(j) \leq ne(i)} rt(\pi_j) \leq dl(\pi_i) \wedge$
30:                 $pr(\pi) \geq \operatorname{argmax} pr(p),\ p \in V$ then
31:                 $\Gamma_s = \Gamma'_s \cup \{\pi\}$
32:             end if
33:         end if
34:     end for
35:     return $\Gamma_p \cup \Gamma_s$
36: end function
```

Scheduling in N-2APL is pre-emptive in that the adoption of a new plan π may prevent previously scheduled plans with priority lower than π (including currently executing plans) being added to the new schedule. Plans that would exceed their deadline are dropped. In the case of obligations, a sanction will necessarily be incurred, so it is not rational for the agent to continue to attempt to discharge the obligation. In the case of goals, it is assumed that the deadline is hard, and there is no value in attempting to achieve the goal after the deadline.

The scheduling algorithm is shown in Algorithm 1. $ne(\pi)$ is the time at which π_i will next execute, $ex(\pi)$ is the time π_i has spent executing up to this point, $dl(\pi)$ is the deadline for π, and $rt(\pi) = et(\pi) - ex(\pi)$ is the remaining execution time of π.

Non-atomic and atomic plans are scheduled separately in Γ_p and Γ_s respectively. The set of candidate plans is processed in descending order of priority. For each plan π, if π is non-atomic an attempt is made to schedule it in parallel with other non-atomic plans in Γ_p (lines 6–10). To determine feasibility for non-atomic plans it is sufficient to check that the plan can be executed by its deadline. If the plan is atomic, it is added to the schedule Γ_s if it can be inserted into the schedule in deadline order while meeting its own and all currently scheduled deadlines (lines 12–32). A set of atomic plans is feasible iff they can be scheduled earliest deadline first [10]. If a non-atomic or atomic plan violates a prohibition of higher priority than the plan, the plan is dropped. The resulting schedule can be computed in polynomial time (in fact, quadratic time) in the size of Π, and (as we show in section 5) is preference-maximal.

4. OPERATIONAL SEMANTICS

In this section, we sketch the operational semantics of N-2APL in terms of a transition system. Each transition transforms one configuration into another and corresponds to a single computation/execution step. In the following subsections, we first present the configuration of individual N-2APL agent programs (henceforth agent configuration) then the configuration of multiagent system programs (henceforth multiagent system configuration), before finally presenting transition rules from which possible execution steps (i.e., transitions) for both individual agents as well as multiagent systems can be derived.

4.1 N-2APL Configuration

The configuration of an individual agent consists of its identifier, beliefs, goals, prohibitions, planning goal rules, procedure call rules, plans, events, and a preference ordering on goals and sanctions. Each plan is associated with the goal and practical reasoning rule that gave rise to the plan (in order to avoid redundant applications of practical reasoning rules, e.g., to avoid generating multiple plans for one and the same goal). It should be noted that the belief base and each goal in the goal base are consistent as only positive atoms are used to represent them.

DEFINITION 1 (AGENT CONFIGURATION). *The configuration of a N-2APL agent is defined as $A_\iota = \langle \iota, \sigma, \gamma, \Pi, \xi, \succ, PG, PC, PR \rangle$ where ι is the agent's identifier, σ is a set of belief expressions $\langle belief \rangle$ representing the agent's belief base, γ is a list of goal expressions $\langle goal \rangle$ representing the agent's goal base, Π is the agent's plan base consisting of a set of plan entries $(\langle plan \rangle, \langle goal \rangle, \langle pgrule \rangle)$ representing the agent's plans together with their next execution times, ξ is the agent's event base containing also elements of the form $prohibition(p, s)$, $\succ$ is the preference ordering, PG is the set of planning goal rules, PC is a set procedure call rules, and PR is a set of plan repair rules.*

Since the agent's practical reasoning rules and preference ordering do not change during an agent's execution, we do not include them in the agent's configuration and use $A_\iota = \langle \iota, \sigma, \gamma, \Pi, \xi \rangle$ to denote an agent's configuration.

The configuration of a multiagent system is defined in terms of the configuration of individual agents and the state of their organisation. The state of the agents' organisation is a set of facts that hold in that organisation.

DEFINITION 2 (MULTIAGENT SYSTEM CONFIGURATION). *Let A_ι be the configuration of agent ι and let χ be the state of the agents' organisation. The configuration of a N-2APL multiagent system is defined as $\langle A_1, \ldots, A_n, \chi\rangle$.*

The initial configuration of a multiagent system is determined by its corresponding multiagent system program and consists of the initial configuration of its individual agents (determined by their corresponding N-2APL programs) and the initial state of their organisation.

DEFINITION 3 (INITIAL CONFIGURATION). *Let ι be the identifier of an agent that is implemented by a N-2APL program. Let σ be the set of $\langle belief\rangle$-expressions specified in the N-2APL program and γ be the list of $\langle goal\rangle$-expressions from the same program. Then, the initial configuration of agent ι is defined as tuple $\langle \iota, \sigma, \gamma, \emptyset, \emptyset\rangle$. Let also χ be a set of facts and $A_1, \ldots, A_n$ be the initial configurations of agents $1, \ldots, n$ that are specified in the multiagent system program. The initial configuration of the multiagent systems is defined as tuple $\langle A_1, \ldots, A_n, \chi\rangle$.*

4.2 Transition Rules

The execution of a N-2APL multiagent program modifies its initial configuration by means of transitions that are derivable from the transition rules given below. Due to lack of space, we do not provide transition rules for the execution of plans, see e.g., [4].

4.2.1 Receiving Detached Norms

As explained in section 2, a normative organisation can broadcast an obligation or a prohibition event to a specific agent. We assume transition rules from which a transition of a normative organisation is derivable, i.e., we assume transition rules that can be used to derive transitions $\chi \xrightarrow{n\text{-}event} \chi'$, where *n-event* is an event such as $obligation(\iota, o, d, s)$ or $prohibition(\iota, p, s)$. Such transition rules specify under which conditions an obligation or a prohibition should be issued for a specific agent. The study of such conditions is out of the scope of this paper and can be found in, e.g., [5]. The following transition rule allows a normative organisation to broadcast normative events (e.g, $obligation(\iota, o, d, s)$ or $prohibition(\iota, p, s)$) and ensures that the events are delivered to the appropriate agent. An obligation event is added to the agent's goal base and a prohibition event is added to its event base.

$$\frac{\chi \xrightarrow{n\text{-}event} \chi'}{\langle A_0, \ldots, A_\iota, \ldots, A_n\rangle \longrightarrow \langle A_0, \ldots, A'_\iota, \ldots, A_n\rangle} \quad (1)$$

where
$A_\iota = \langle \iota, \sigma, \gamma, \Pi, \xi\rangle$,
$A'_\iota = \langle \iota, \sigma, \gamma \cup \{o:d\}, \Pi, \xi, \succ'\rangle$ if *n-event* $= obligation(\iota, o, d, s)$,
$A'_\iota = \langle \iota, \sigma, \gamma, \Pi, \xi \cup \{prohibition(p, s)\}, \succ'\rangle$ if *n-event* $=$ *prohibition*(ι, p, s).

4.2.2 Planning Goal Rules

A N-2APL agent generates plans by applying PG-rules of the form $\kappa \leftarrow \beta \mid \pi : t$. An agent can apply one of its PG-rules $\kappa \leftarrow \beta|\pi : t$, if κ is entailed (with some substitution τ_1) by one of the agent's goals, namely by some g such that $g : d \in \gamma$, β is entailed (with substitution $\tau_1\tau_2$) by the agent's belief base, and there is no plan in the plan base that has been generated (and perhaps partially executed) by applying the same PG-rule to achieve the same goal. Applying the PG-rule $\kappa \leftarrow \beta \mid \pi : t$ attempts to add $\pi\,\tau_1\tau_2$ with deadline d and execution time t to the agent's plan base.

$$\frac{\begin{array}{c}\exists\,(g\!:\!d) \in \gamma : g \models_g \kappa\tau_1 \;\&\; \sigma \models \beta\,\tau_1\tau_2 \\ \& \neg\exists \pi' \in P : (\pi', g\!:\!d, (\kappa\tau_1 \leftarrow \beta \mid \pi\!:\!t)) \in \Pi\end{array}}{\langle \iota, \sigma, \gamma, \Pi, \xi\rangle \longrightarrow \langle \iota, \sigma, \gamma, \Pi', \xi\rangle} \quad (2)$$

where τ_1, τ_2 are substitutions, P is the set of all possible plans and $\Pi' = \text{SCHEDULE}(\Pi \cup \{(\pi\,\tau_1\tau_2,\ g : d,\ (\kappa\tau_1 \leftarrow \beta \mid \pi : t))\}, prohibitions(\xi))$. Here, SCHEDULE is defined as in Algorithm 1 and *prohibitions* is a function that takes a set of events and returns all prohibition events in that set.

4.2.3 Concurrent Plans

The applications of planning goal rules may generate both atomic and non atomic plans. External actions in different non-atomic plans can be executed in parallel. Atomic plans are executed in sequence, rather than in parallel. Actions within each plan are executed in strict sequence, i.e., the next action in the plan is not executed until the previous action has completed execution or failed. We say a plan is *executable* if its next execution time is *now* and the previous action in the plan has successfully completed execution, otherwise it is not executable (i.e., an action initiated at a previous deliberation cycle is still executing, or has failed, or the plan is atomic and scheduled for execution at a later time). Execution of internal actions is assumed to occur in the main interpreter thread, whereas execution of external actions is assumed to occur in separate threads (otherwise the actual execution time of a plan π would bear little relation to its expected execution time, $et(\pi)$) .

An agent executes its plans concurrently by interleaving the execution of the next action (or the initiation of the execution of the action in a separate thread in the case of external actions) of all executable plans whose next execution time is *now*

$$\frac{\langle \iota, \sigma, \gamma, \rho, \xi\rangle \longrightarrow \langle \iota, \sigma', \gamma', \rho', \xi'\rangle}{\langle \iota, \sigma, \gamma, \Pi, \xi\rangle \longrightarrow \langle \iota, \sigma', \gamma', \Pi', \xi'\rangle} \quad (3)$$

where ρ is executable and $\Pi' = (\Pi \setminus \rho) \cup \rho'$.

5. NORM-AWARE RATIONALITY

In this section, we justify the adoption of a preference-maximal set of plans as an appropriate standard of rationality for a norm-aware agent. We show that key assumptions underlying the design of N-2APL are necessary in the sense that if they are relaxed, a N-2APL agent is either no longer a norm-aware rational agent, or its deliberation about goals, norms and sanctions is intractable.

In what follows, we assume that the agent's plans have a positive expected execution time and a deadline, and that atomic plans are scheduled on a single processor. We also assume that plans and prohibitions have priorities, which are either ordered by a total preference pre-order, or have numerical values. The definition of a feasible set of plans is given in section 3.3. Note that in order to be able to establish whether a set of plans is feasible, the agent needs to know expected execution times of the plans.

We call a set of plans *optimal* if it is feasible and has maximal utility; namely, if the preferences over goals and sanctions are ordered by a preference pre-order, then it contains the highest number of the high priority plans; if priorities are numerical, then their sum is maximal.

DEFINITION 4. *An agent is* perfectly norm-aware rational *if it commits to an optimal set of plans.*

The problem of finding an optimal schedule is NP-complete (e.g., the special case in which all tasks have the same deadline can be reformulated as a 0-1 knapsack problem [7]).

Since the problem of scheduling an optimal set of plans is not tractable, we have the following theorem:

THEOREM 1. *A perfectly norm-aware rational agent cannot have a tractable deliberation procedure.*

In this paper, we assume a weaker definition of norm aware rationality. This definition relies on the notion of a preference-maximal set of plans given in Section 3.3 and requires that preferences are ordered on an ordinal scale. Note that in this case every optimal set of plans is preference-maximal, but not vice versa. For example, for three atomic plans with the same priority, π_1, π_2 and π_3, such that π_1 is not feasible with either of π_2 or π_3 and $\{\pi_2, \pi_3\}$ is feasible, the only optimal set is $\{\pi_2, \pi_3\}$. However, $\{\pi_1\}$ is a preference-maximal set of plans.

DEFINITION 5. *An agent is* norm-aware rational *if it commits to a preference-maximal set of plans.*

The set of goals achieved by a *norm-aware rational* agent is determined by its program. If the program is such that the belief contexts of PG rules are disjoint, then the set of goals achieved by the successful execution of a preference-maximal set of plans is also 'preference-maximal', in the sense that the agent will only fail to achieve a goal if it has no applicable plan for a goal, or the plan to achieve the goal is not feasible together with the plans to achieve goals of the same or higher priority.

THEOREM 2. *A N-2APL agent is norm-aware rational and its deliberation procedure is tractable.*

PROOF. We show that the N-2APL scheduling algorithm returns a preference-maximal schedule; tractability is obvious. The N-2APL scheduling algorithm builds two separate schedules, a parallel and a sequential one. The set of plans Γ_p in the parallel schedule contains all non-atomic plans which are individually feasible (the time remaining to their deadline is greater than their expected execution time) and do not violate prohibitions of higher priority. This set of plans is clearly maximal given the prohibitions. Note that the feasibility of Γ_p is not affected by the membership of the sequential schedule Γ_s and vice versa.

For the sequential schedule, the algorithm constructs a sequence of sets starting with $\Gamma_0 = \emptyset$, and sets Γ_i to be $\Gamma_{i-1} \cup \{\pi_i\}$, if $\Gamma_{i-1} \cup \{\pi_i\}$ is feasible in deadline order, or Γ_{i-1} otherwise. The last set Γ_n is Γ_s. By construction, Γ_s is a feasible set of plans. Γ is also clearly a maximally feasible subset of Π: there is no atomic $\pi \in \Pi$ such that $\pi \notin \Gamma_s$ and $\Gamma_s \cup \{\pi\}$ is feasible. To prove that it is preference-maximal, let $\pi_i \in \Pi$, $\{\pi_i\}$ feasible, and $\pi_i \notin \Gamma_s$. We need to show that π_i is incompatible with some subset of Γ_s which contains only plans of the same or higher priority, or is incompatible with a higher priority prohibition. Since the plans are added to Γ_s in descending order of priority, when π_i is considered and found incompatible with Γ_{i-1}, the priority of π_i is at most the lowest priority in Γ_{i-1}. $\square$

In the rest of this section, we show that the assumptions we made concerning the normative system and the agent programming language semantics are essential to guarantee tractability of a norm aware rational agent's deliberation.

THEOREM 3. *If a normative system has prohibitions with real-time deadlines, then a norm-aware rational agent cannot have a tractable deliberation procedure.*

PROOF. If prohibitions have real-time deadlines, the scheduling problem is equivalent to the 'sequencing with release times and deadlines' problem (SRTD), which is known to be strongly NP-complete [7]. SRTD is intractable in the sense that it admits no bounded approximation computable in time polynomial in the problem size and the bound (unless $P = NP$). $\square$

N-2APL also places certain restrictions on the syntax of 2APL programs, and changes the semantics of key constructs such as the non interleaving (atomic) operator. If these assumptions are relaxed, the agent is no longer norm-aware rational or deliberation about norms is intractable. We now make these assumptions precise.

THEOREM 4. *If an agent's plans may have atomic sub-plans the agent is not norm-aware rational, or its deliberation procedure is intractable.*

PROOF. Note that if an agent's plans may contain atomic sub-plans $\pi = \pi_1; [\pi_2]; \pi_3; [\pi_4]$, the agent must schedule the atomic sub-plans $[\pi_2]$ and $[\pi_4]$ together with its other atomic plans, but subject to the constraint that the preceding non-atomic sub-plans π_1 (in the case of $[\pi_2]$) and π_2 (in the case of $[\pi_4]$) of π have finished executing. If information about the execution times of each atomic and non-atomic sub-plan is not available, the agent is not norm-aware in the sense that it may commit to plans that it subsequently has to abandon, even if its goals and norms do not change. In effect, the agent is unable to tell when the execution of an atomic sub-plan may have to be scheduled, and so may adopt a non preference-maximal schedule. If, on the other hand, we assume that information about the execution times of each plan segment is available (and is used in scheduling), the scheduling problem is again equivalent to SRTD and is intractable. $\square$

THEOREM 5. *If atomic plans cannot be executed in parallel with non-atomic plans, an agent is not norm-aware rational or its deliberation procedure is intractable.*

PROOF. If we adopt the 2APL semantics for the non interleaving operator,[3] i.e., the execution of an atomic plan should not be interleaved with the execution of any other plan, we again get a combinatorial scheduling problem because non-atomic plans can be split in various groups depending on their end times. This problem reduces to the batch scheduling problem which is NP-hard [12]. $\square$

THEOREM 6. *If external actions cannot be executed in parallel, an agent is not norm-aware rational or its deliberation procedure is intractable.*

PROOF. If external actions are interleaved, the expected execution time of a plan is dependent on the other plans in the agent's schedule. If the execution times of each external action in a plan are not known, the resulting schedule is not guaranteed to be preference-maximal. If the execution time of each external action are known (and is used in scheduling), the scheduling problem is again reducible to SRTD, for both atomic and non-atomic schedules. $\square$

6. RELATED WORK

Our notion of norm-awareness is related to, e.g., [14], where it is argued that the ability of agents to reason about the norms of an organisation in which they operate is crucial for their decisions to enter and leave organisations or to respect/violate norms.

There has been considerable recent work on approaches to programming normative systems. The JaCaMo programming framework combines the Jason [2], Cartago [13], and MOISE$^+$ [8] platforms. In this integrated approach, the organisational infrastructure of a multiagent system consists of organisational artefacts and

[3]This semantics is also used by Jason.

agents that together are responsible for the management and enactment of the organisation. An organisational artefact employs a normative program which in turn implements a MOISE$^+$ specification. A programming language for the implementation of normative programs as well as a translation of MOISE$^+$ specifications into normative programs is described in [9]. JaCaMo allows Jason agents to interact with organisational artefacts, e.g., to take on a certain role. (As the idea of organisational artefacts based on normative programs is closely related to the 2OPL architecture for normative systems [5] used in this paper, we believe that our N-2APL agents could also be used in the JaCaMo framework.) In contrast to N-2APL, the Jason agents in this combined model have no explicit mechanisms to reason about norms (obligations and prohibitions) and their deadlines and sanctions in order to adapt their behaviour at run time. Another approach that integrates norms in a BDI-based agent programming architecture is proposed in [11]. This extends the AgentSpeak(L) architecture with a mechanism that allows agents to behave in accordance with a set of non-conflicting norms. As in N-2APL, the agents can adopt obligations and prohibitions with deadlines, after which plans are selected to fulfil the obligations or existing plans are suppressed to avoid violating prohibitions. However, unlike N-2APL, [11] does not consider scheduling of plans with respect to their deadlines or possible sanctions.

There are also several agent languages which incorporate deadlines, including the Soft Real-Time Agent Architecture [16] and AgentSpeak(XL) [1]. These architectures use the TÆMS (Task Analysis, Environment Modelling, and Simulation) framework [6] together with Design-To-Criteria scheduling [17] to schedule intentions. TÆMS provides a high-level framework for specifying the expected quality, cost and duration of of methods (actions) and relationships between tasks (plans). Like N-2APL, tasks (and methods) can have deadlines, and TÆMS assumes that the expected execution times (and quality and costs) of tasks are available. As in N-2APL DTC can produce schedules which allow interleaved or parallel execution of tasks. However the view of 'deadline' used in these systems is different from that used here, in that deadlines are not hard (tasks still have value after their deadline), and they provide no support for normative concepts such as obligations, prohibitions and sanctions. To the best of our knowledge, they have not been used to develop norm-aware agents. AgentSpeak(RT) [15] is a version of AgentSpeak(L) which allows the specification of deadlines and priorities for tasks. However, as with SRTA and AgentSpeak(XL) it provides no support for normative concepts.

7. CONCLUSIONS

We have presented N-2APL, a programming language for norm-aware agents. N-2APL provides support for obligations with deadlines, prohibitions, and sanctions. N-2APL agents are guaranteed to be norm-aware rational, that is, to commit to a set of plans of the highest priority which do not violate higher priority prohibitions, and which are feasible. We believe that N-2APL represents a good compromise in the design of a norm-aware agent programming language. If the key assumptions underlying the design of N-2APL are relaxed, an agent either no longer satisfies norm-aware rationality, or its deliberation about goals, norms and sanctions is intractable. Although we have developed our ideas in the context of the 2APL agent programming language, the extensions to support norm-aware deliberation could be incorporated in a straightforward way to other BDI-based agent programming languages.

8. REFERENCES

[1] R. Bordini, A. L. C. Bazzan, R. d. O. Jannone, D. M. Basso, R. M. Vicari, and V. R. Lesser. AgentSpeak(XL): Efficient intention selection in BDI agents via decision-theoretic task scheduling. In *Proc AAMAS'02*, pages 1294–1302, 2002.

[2] R. H. Bordini, M. Wooldridge, and J. F. Hübner. *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons, 2007.

[3] J. Broersen, M. Dastani, J. Hulstijn, and L. van der Torre. Goal generation in the BOID architecture. *Cognitive Science Quarterly*, 2(3-4):428–447, 2002.

[4] M. Dastani. 2APL: a practical agent programming language. *Autonomous Agents and Multi-Agent Systems*, 16(3):214–248, 2008.

[5] M. Dastani, D. Grossi, J.-J. C. Meyer, and N. Tinnemeier. Normative multi-agent programs and their logics. In *Proc. KRAMAS'09*, LNCS 5605, pages 16–31, 2009.

[6] K. S. Decker and V. R. Lesser. Quantitative modeling of complex environments. *International Journal of Intelligent Systems in Accounting, Finance and Management*, 2:215–234, 1993.

[7] M. R. Garvey and D. S. Johnson. *Computers and Intractability*. W. H. Freeman and Company, New York, 1979.

[8] J. Hübner, J. Sichman, and O. Boissier. $\mathcal{S}-\mathcal{M}$OISE$^+$: A middleware for developing organised multi-agent systems. In *Proc. COIN'06*, LNCS 3913, pages 64–78. Springer, 2006.

[9] J. F. Hübner, O. Boissier, and R. H. Bordini. From organisation specification to normative programming in multi-agent organisations. In Proc. CLIMA XI, LNCS 6245 , pages 117–134. Springer, 2010.

[10] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *JACM*, 20(1):46–61, 1973.

[11] F. R. Meneguzzi and M. Luck. Norm-based behaviour modification in BDI agents. In Proc. AAMAS'09, pages 177–184. IFAAMAS, 2009.

[12] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall, 2002.

[13] A. Ricci, M. Viroli, and A. Omicini. Give agents their artifacts: the A&A approach for engineering working environments in MAS. In Proc. AAMAS'07, IFAAMAS, 2007.

[14] M. B. van Riemsdijk, K. V. Hindriks, and C. M. Jonker. Programming organization-aware agents: A research agenda. In *Proc. ESAW'09*, LNAI 5881, pages 98–112. Springer, 2009.

[15] K. Vikhorev, N. Alechina, and B. Logan. Agent programming with priorities and deadlines. In Proc. AAMAS'11, pages 397–404, 2011.

[16] R. Vincent, B. Horling, V. Lesser, and T. Wagner. Implementing soft real-time agent control. In *Proc. AGENTS'01*, pages 355–362, 2001.

[17] T. Wagner, A. Garvey, and V. Lesser. Criteria-directed heuristic task scheduling. *International Journal of Approximate Reasoning*, 19:91–118, 1998.

Metamodel-Based Metrics for Agent-Oriented Methodologies

Noélie Bonjean[*]
bonjean@irit.fr

Antonio Chella[†]
antonio.chella@unipa.it

Massimo Cossentino[‡]
cossentino@pa.icar.cnr.it

Marie-Pierre Gleizes[*]
gleizes@irit.fr

Frédéric Migeon [*]
migeon@irit.fr

Valeria Seidita[†]
valeria.seidita@unipa.it

ABSTRACT

A great number of methodologies has been already introduced in the agent-oriented software engineering field. Recently many of the authors of these methodologies also worked on their fragmentation thus obtaining portions (often called method or process fragments) that may be composed into new methodologies. The great advancement in this field, however does not correspond to equivalent results in the evaluation of the methodologies and their fragments. It is, for instance, difficult to select a fragment in the composition of a new methodology and to predict the methodology's resulting features. This work introduces a suite of metrics for evaluating and comparing entire methodologies but also their composing fragments. The proposed metrics are based on the multi-agent system metamodel. The metrics have been applied to the ADELFE and PASSI methodologies, results prove the usefulness of the proposed approach and encourage further studies on the matter.

1. INTRODUCTION

The interest for the concept of agent as the composing element of an autonomous system, capable of interacting with other agents in order to satisfy its design objectives, has grown since the 1980s. Nowadays, a great number of Agent Oriented Methodologies[1] (AOM) have been proposed [1]. These methodologies focus on different aspects and, offering different functionality with different levels of detail, address a scale of Multi-Agent Systems (MAS). The diversity of approaches offers rich resources for developers to draw on, but can also be a hindrance to progress if their commonalities and divergences are not readily understood. Moreover, an attempt from the wide range of AOMs is to benefit from different methodologies by combining their particular features. Several methodologies have been already combined before they were split up into fragments in order to adapt and/or design new methodologies [16]. It is therefore necessary to provide a way of comparison that would help to choose the suited method. In the last few years, the evaluation of MAS software engineering techniques has been one of the most active areas of research and some comparison frameworks have been proposed taking into account what concepts are manipulated, notations used, and development process or pragmatics adopted [2] [23]. Despite this, there is no objective, complete and systematic way to evaluate MAS development methods and tools. Besides, it is difficult to select a fragment in the composition of a new methodology and to predict the methodology's resulting features [8]. In this work, we propose some metrics measuring relevant features of AOM that deal with objective criteria for evaluating and comparing entire methodologies but also their composing fragments.

All methods of evaluation are influenced by the same factors, some affecting the evaluation of the structural features of the methodology itself, some others its enactment performances. The evaluation of static aspects is mainly influenced by the appreciation of: the importance given to designer's experience (a huge availability of guidelines, for instance, minimize that), the work to be done, the artefacts to be produced. These elements correspond to the typical triangle stakeholder-activity-work product considered as central by many design process composition and modelling approaches (for instance SPEM [18]). As regards methodology enactment, factors like problem complexity, designers' experience/number and the development context are crucial. All these variables are connected together. The study of all the variables at the same time is far too complex. Different approaches have proposed separate studies of some of these variables, for instance [4][22][9][15][10].

In the proposed approach, we base the evaluation on the assumption that the MAS metamodel (MMM hence after) adopted in a methodology directly influences the three crucial elements of the methodology (stakeholders, activities, work products) and it is conversely influenced by them. As a result, we think that some useful indications about the methodology features may be obtained by the application of metrics based on the MMM.

For this reason, this work starts analysing the metamodel

[*] *Institut de Recherche en Informatique de Toulouse - IRIT Université Paul Sabatier, Toulouse, France*

[†] *Dipartimento di Ingegneria Chimica Gestionale Informatica Meccanica Università degli Studi di Palermo, Italy*

[‡] *Istituto di Reti e Calcolo ad Alte Prestazioni, Consiglio Nazionale delle Ricerche - ICAR/CNR Palermo, Italy*

[1] In this paper we consider the term methodology and design process as synonyms

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

of the fragments obtained by the fragmentation of the methodology. Such fragments (sometimes addressed as process fragments, method fragments or chunks) constitute the root constructs of the methodology itself and they have been extracted by considering a precise granularity criterion: each group of activities (composing the fragment) should significantly contribute to the production/refinement of one of the main artefacts of the methodology (for instance, a diagram or a set of diagrams of the same type). Following this assumption, fragments obtained from different methodologies are based on a similar level of effort (although with some obvious differences among them) and produce results of similar complexity (a work product like a diagram). The approach is also based on some other assumptions we made: (1) The fragments included in a possible loop of methodology process are counted only one time. This ensures that the different methodologies (and related fragments) can be fairly compared. (2) Finally, in order to be comparable, some rules have to be followed in the description of the fragment metamodel. Such rules ensure that different methodologies (and composing fragments) refer to metamodels described with a comparable level of detail.

The proposed metrics are: **fragment input** (measuring the dependency of a fragment on the others), **fragment output** (measuring the complexity of the output work product), **fragment creative effort** (measuring the complexity of the design effort spent on the fragment), **fragment freedom degree** (measuring the degree of freedom offered to the designer while working in the fragment activities). The metrics have been applied to the ADELFE [20] and PASSI [6] methodologies. The results have been used to validate the proposed metrics.

The paper is organized as follows: details about the definition of MAS metamodels and the process fragments are introduced in section 2. In section 3, the proposed metrics are defined specifying the evaluation criteria and the formulas that allow obtaining quantitative results. Section 4 shows and discusses some results coming from the application of the metrics to two methodologies. Some works are related in section 5 before concluding with some prospects.

2. BACKGROUND

Before discussing the proposed metrics we describe all the constructs used for their creation and their application. In this section we first introduce the concept of MAS metamodel and what is its importance in a design process, how we model a design process using the IEEE FIPA styles [13] also including the concept of fragments and finally which diagrams are useful for applying the proposed metrics.

2.1 Multi-Agent System Metamodel

Metamodelling techniques lie in creating a set of concepts used for describing the properties of models. In the same way a model is an abstraction of real world's elements, phenomena, etc. , a metamodel is a further abstraction of a model. A model is always into compliance with its metamodel that rules the way in which a model has to be constructed.

Metamodelling is an essential foundation of Model Driven Engineering (MDE) proposed by OMG[2] where the language for describing metamodels is Meta Object Facility (MOF) [17].

The traditional modelling infrastructure proposed by OMG is made by four layers each one characterised by an *instance_of* relationship with the above layer. The bottom level is the level M0, it contains the user data and is called the **instance model**. The system solving a specific problem is that running on a platform, it represents the elements that exist while the system runs on the real-world platform and manages the user data. In this work, a system corresponds to a MAS. The instance model is created by instantiating what is held in the so called **user model** (level M1); level M2 contains the model of information for instantiating the M1 models and for this purpose it is called **metamodel layer**. Finally level M3 contains the **model information** for creating metamodels; hence the meta-metamodels that is usually reported as MOF.

The MAS metamodel we consider for this work is the one defined in [7]. It presents a multi-layer structure in the same way of OMG and is composed of constructs. We define a *construct* as a general term referring to one of the following entities: *elements*, *relationships*, *attributes* and *operations*. We therefore define a MAS Metamodel Construct (**MMMC**) as one of the previously mentioned entity of the metamodel.

A MAS Metamodel Element (**MMME**) is an entity of the metamodel (M2 level) that is instantiable into an entity of the system model (M1 level). Examples of MMME are: classes, use cases,... Such elements may be instantiated in the corresponding model element.

A MAS Metamodel Relationship (**MMMR**) is the construct used for representing the existence of a relationship between two (or more) instances of MMMEs. For instance, the aggregation relationship among two instances of a MMME class is an instance of the MMMR association.

A MAS Metamodel Operation (**MMMO**) is a behavioural feature of a classifier that specifies the name, type, parameters, and constraints for invoking an associated behaviour.

A MAS Metamodel Attribute (**MMMA**) is a particular kind of elements used for adding properties to MMMEs. A MMMA is a structural feature and it relates an instance of the class to a value or collection of values of the type of the attribute. The attributes type is a MMME. It is worth to note that several metamodels do not list any MMMA. This is the consequence of the fact that, according to MMMA definition, an attribute is a relationship between the class and another element of the metamodel. It is a choice of the designer to represent such relationship using attributes or other strategies (an explicit relationship with the owned element). In order to support all metamodelling style, MMMAs are included in the set of MMMCs supported by our approach.

We claim that every design process is underpinned by a system metamodel, therefore each time a designer is developing her/his MAS (s)he has at his disposal a set of elements, relationships, attributes and operations (s)he can manage for creating the system models; hence for drawing all the work products composing the system model.

In order to provide means for understanding the metrics presented in this work, in the following subsections we will provide an overview on the modelling actions made on the system metamodel constructs during design and the mutual relationships between system metamodel and work product by means of what we call *wp_content diagram*.

[2]http://www.omg.org/index.htm

2.2 Modelling Actions

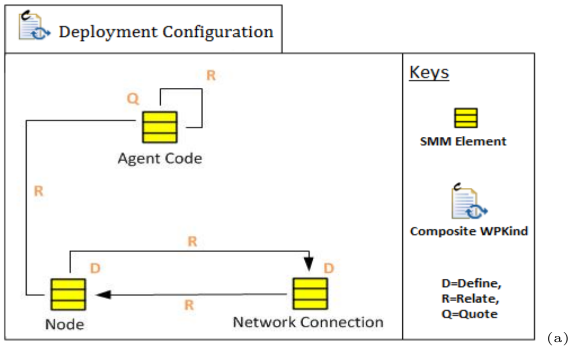
While composing a work product four different kinds of action can be made on the metamodel constructs; the designer may:

- Instantiate an element in the work product. This corresponds to create a new model element. This modelling action is labelled **Define** (D).
- Instantiate a relationship in the work product. This corresponds to create a new relationship among two model elements according to what is permitted by the metamodel. This modelling action is labelled **Relates** (R).
- Refine an already defined element in the work product. This corresponds to instantiate a new attribute or operation. Refining an element corresponds to performing the following modelling actions: **Refine Attribute** (RFA) or **Refine Operation** (RFO). We note that in both cases it includes the quotation of that element.
- Use an already defined construct in the work product. New work products often relate the new elements to other elements that have been introduced in previous process activities and therefore reported in their corresponding output work products. Reporting an already defined metamodel construct corresponds to performing the following modelling actions: **Quote** (Q), **Quote Relationship** (QR), **Quote Attribute** (QA) or **Quote Operation** (QO).

2.3 Fragment and Relevant Diagrams

When we talk about "portion of work" we explicitly refer to the way of representing design processes established by IEEE FIPA [13] that founds on some underlying ideas: design process is a set of activities performed in order to reach design goals and the whole design process can be divided in phases which in turn are composed of activities and tasks. Each portion of work is devoted to deliver work products, for instance activity is devoted to produce a finer grained artefact, one single work product like a diagram possibly complemented by a text description whereas phase delivers major artefacts, for instance requirements analysis documents. Therefore design process is composed of portion of works to be performed by stakeholders (or process role) in order to deliver work products (models of the system). Each work product represents a set of elements of the whole system metamodel the design process underpins. Moreover a portion of work is usually referred to as "fragment" [12][3][5][19]. The way to describe fragment is still a work in progress and descends from the approved IEEE FIPA standard on process documentation.

What is important now is that the system metamodel assumes a very central role in designing and it is the most important design process component enabling, among others, tracking all the transformations and actions designer does while producing a work product. Besides we have to consider that design process can be seen as a sequence of fragments. In order to calculate the metrics that we propose in this paper it is important to have a look to two important diagrams used in the documentation of design process, hence of each fragment composing it. They are the *Work Product Content diagram* and the *System Metamodel diagram*.



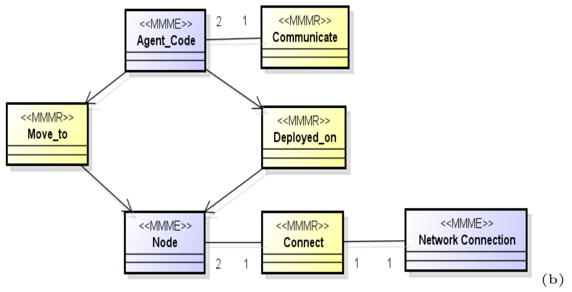


Figure 1: An Example of the Workproduct Content Diagram and of MAS metamodel diagram

The first is a specific kind of diagram (see [7] for more details) devoted to collect all the metamodel constructs that are managed during the design process enactment and are also reported in the work product. It represents the relationships between each work product produced during the design process and all the constructs of the metamodel that are here drawn. An example is given in Fig.1.(a) showing the elements and the relationships between them. They are footnoted with a specific label that represents the kinds of action made on it. The labels correspond to the modelling actions presented previously (cf. 2.2). In addition, the package with an icon on the left uppermost corner points out the work product and its kind (see [21] for the complete list and an explanation on work product kinds).

The second diagram is the MAS Metamodel diagram that shows all the system metamodel constructs that are managed by the designer in using a specific design process. This also includes all the constructs that are accepted as external inputs of the overall process whereas in the work product content diagram only the constructs reported in the work product are shown. Fig.1.(b) shows the MAS Metamodel of the Communication Ontology Description of PASSI.

3. FRAGMENT METRICS

As opposed to object-oriented methodologies, there has not been much work in comparing agent-oriented methodologies because of the intrinsic features in different MASs and their application context. There is therefore a real difficult in evaluating them. The following section presents a set of criteria based on the agent-oriented metamodel of each process fragment. Actually, four metrics are defined: fragment input, fragment output, fragment creative effort and fragment freedom degree.

3.1 Fragment Input

The Fragment Input (FI) is an architectural metric. It represents the input data required by a fragment i.e. a kind of constraints representing a guard condition. It measures the dependency of a fragment to another in a method. Actually, the input of a fragment is the number of its immediately needed constructs. High fragment input identifies fragments with a relevant dependency on other portions of the design process. Besides, a low fragment input is desirable for relating fragments because the probability to find needed constructs is higher. Thus, a fragment can be more reused, which is usually a good objective. For instance, it is common to find fragments with a low FI at the beginning of the process while this number is often higher going on inside the lyfe-cicle.

Let F be all fragments from a design process and I_f be the set of constructs required by the fragment f. We calculate:

$$\forall f \in F, FI(f) = |I_f|$$

Usually, the fragment input shows the most specific fragments of a process, i.e. the fragments that are related each other by tight dependency relationships.

3.2 Fragment Output

The Fragment Output (FO) measures the work product complexity of the output constructs. Actually, work product complexity represents the complexity in reading and understanding a work product. We think this is mainly affected by two numbers: the number of different types of MMMC that can be represented in the work product (according to what specified in the process) and the number of instances of these MMMC actually represented in a specific work product. This latter number belongs to the M1 level of representation and therefore it is out of the scope of our study. The previous number is composed by the number of the different MMMC that, in the specific work product, may be represented, whatever action the designer performs on them in the fragment.

Let F be all fragments from a design process and O_f be the set of constructs in the fragment f. The constructs can be instantiated or quoted or refined. We calculate:

$$\forall f \in F, FO(f) = |O_f|$$

The fragment output shows the most complex fragments of the process. Actually, the more components are outputted in a fragment the more complex is to understand the fragment.

3.3 Fragment Creative Effort

The Fragment Creative Effort (FCE) is a part of the complexity design effort. Actually, the complexity design effort depends on the specific problem, the skills of the designers and the used design process. The problem complexity is inconveniently informal and heterogeneous and it is treated implicitly by folding it into the solution design in software engineering approach. Because some engineers are multidisciplined, process design is divided into different phases. Process design however, is not a formula exercise and as a engineer in any discipline, once the designer has familiarized her/himself with these skills, (s)he will be able to develop her/his own system. Therefore, the skills and knowledge of an engineer influence the design effort of the produced system. Measuring this influence is a complex task that will be studied later. In this first method evaluation, we do not take into account the designer profile, we suppose a unique user. Finally, the analysing method and more specifically the meta-model provides a part of the complexity design effort. Therefore, the fragment creative effort measures the effort done by the design in performing the definition of the portion of system related to a specific fragment. This effort is related uniquely to the introduction of new elements, relationships, attributes and operations in the system model.

The fragment creative effort is measured for each fragment. It is the number of defined or related or refined constructs in a fragment. Let F be all fragments from a method and D_f be the set of defined or related or refined constructs in the fragment f, then:

$$\forall f \in F, FCE(f) = |D_f|$$

The fragment creative effort shows the most complex fragments of the method in term of design. This metric is strongly related to the fragment output but while the FCE addresses the creative part of the work, the FO measures the complexity of the whole resulting work product. In other words, no contribution to FCE comes, for instance, from constructs reported unchanged from previous portions of work, but these constructs are counted in the FO. As a consequence, this formula expresses the relationship between the two:

$$\forall f \in F, FO(f) \geq FCE(f)$$

3.4 Fragment Freedom Degree

The Fragment Freedom Degree (FFD) represents the degree of freedom granted to designer for a fragment. It is calculated by the ratio of the fragment creative effort over the fragment input. This metric enables to define for a fragment if the introduction of the new constructs is strongly conditioned or not. In a method, high fragment freedom degree is ideally required in the first fragments of the method. In fact, these fragments usually imply designers' own mind and creativity. Then, the fragment freedom degree of the following fragments progressively decreases until to obtain a low fragment freedom degree for the last fragments. Actually, the designer is generally strongly conditioned at the end of the process because during the life-cycle, (s)he is guided in order to converge towards a well defined system.

Let F be all fragments from a design process, then:

$$\forall f \in F, FFD(f) = FCE(f)/FI(f)$$

The fragment creative effort shows the process structure. The ideal process structure might present a high value of FFD at the beginning of the process and a progressive decade of it to a low ration.

4. EXPERIMENTAL RESULTS

Currently, these metrics are evaluate on several Agent-Oriented Methodologies: ADELFE, PASSI, INGENIAS [14] and TROPOS [11]. In this section we present the results obtained for every fragment and gathered by ADELFE and PASSI. We also study the results comparatively for the two methods. This experiment enables us to give an analysis on the intrinsic relevancy of the metrics in subsection 4.1. We

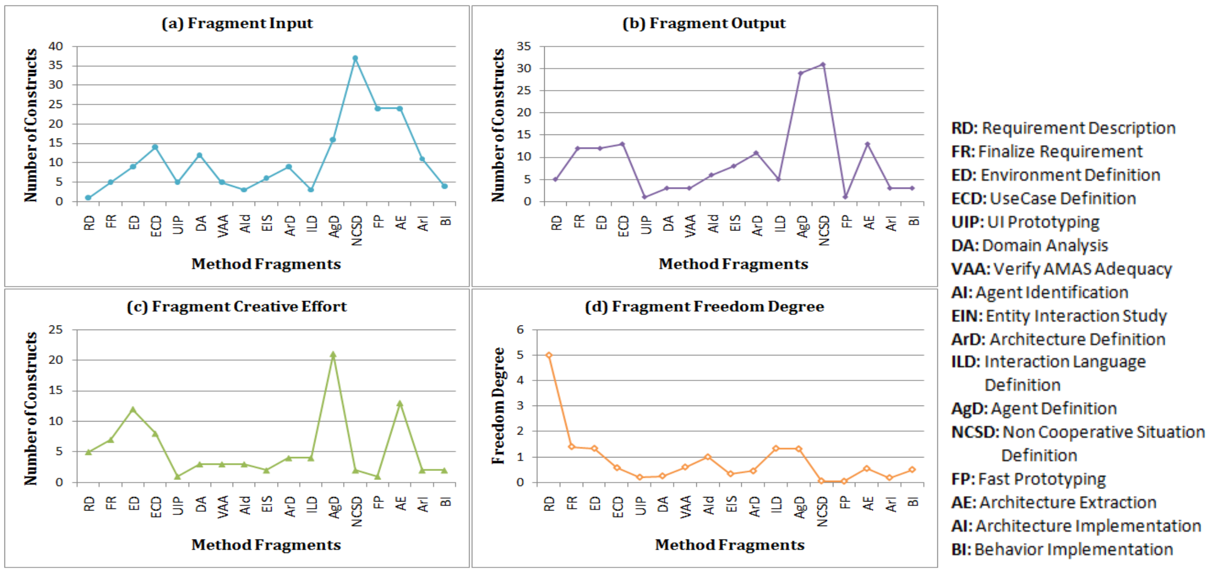


Figure 2: Evaluation of ADELFE

see in subsection 4.2 how process evaluation and fragment evaluation become both possible but are quite different. Finally, we discuss the metrics and the factors that could impact their results.

4.1 Metric Relevancy

It is firstly important to measure the indicators' intrinsic quality and see what kind of correlated analysis they enable. As mentioned in section 3.1, FI gives an idea of the dependencies of each fragment one with another. Therefore, it is natural to find in Fig.2.(a) that higher values are at the ended fragments. It shows their specificity in the ADELFE process and that they will probably induce a difficult reuse. In the same way, for PASSI, we can see in Fig.3.(a) that the first and the fifth fragments (*Domain Requirement Description* and *Domain Ontological Description*) present the lowest values. This is explained by the fact that both analyse the problem domain in terms of the requirements it provides (the first) and of its ontological representation (the second); for this reason, the designer faces a kind of work which is not constrained by other metamodel constructs but it is the result of her/his own observation and reasoning.

Concerning the FO metric, the high values correspond to fragments deliverying heavy weight models of the system. Usually they correspond to the most significant fragments in the process. The usefulness of the fragment low value within the process might be reappraisal. In the ADELFE results in Fig.2.(b), the greatest values are obtained for *Agent Definition* and *Non Cooperative Situation Definition* which indeed are the most significant activities in ADELFE. It is also interesting to note that all fragments settled in the design phase and in the implementation phase (from *Architecture Definition* to *Behaviour Implementation*) have a rather important value except for *Interaction Language Definition* fragment and *Fast Prototyping* fragment. They are two singularities comparatively with the surrounding fragments. It clearly shows the lack in the metamodel of elements to take these activities into account with as much quality as other design or implementation fragments. The PASSI curve reveals an interesting confirmation of these considerations, for instance in the presence of the maximum value for the *Single Agent Structure Definition* fragment which is quite intuitive for an AOSE design process.

The FCE metric also reveals itself to be a very relevant indicator. For ADELFE, three peaks are distinctly drawn above the average value (which is of five constructs) on the FCE plot (cf. Fig.2.(c)). These three peaks correspond to the definition of the most important constructs in ADELFE where the highest efforts are provided to define the environment (*Environment Definition* fragment), the cooperative agent behaviour (*Agent Definition* fragment) and the architecture (*Architecture Extraction* fragment). We will address this discussion in a following subsection. In PASSI, we can note low values of creative effort (cf. Fig.3.(c)) in the *Single Agent Behaviour Description*, in the *Code Reuse* and in the *Code Production*. We were expecting to find these values because these fragments are in the implementation phase of the design process so, because of the nature of PASSI that commits a great effort and a lot of work in the design phase, all the information needed for defining the implementation constructs can be inferred from the previous phase without spending too much effort. Besides another result we imagined is that regarding the *Code Production* and *Code Reuse* creative effort, although these two fragments presents the same value of FO and both the two deals with code concerns, *Code Reuse* has a higher creative effort. During *Code Reuse*, the designer analyses and then reuses patterns of agent and this activity is obviously more demanding than simply writing code on the base of previously drawn structural and behavioural diagrams.

With our experimentation on ADELFE and PASSI, we found these metrics very relevant to show the complexity and the specificity of evaluated methodologies.

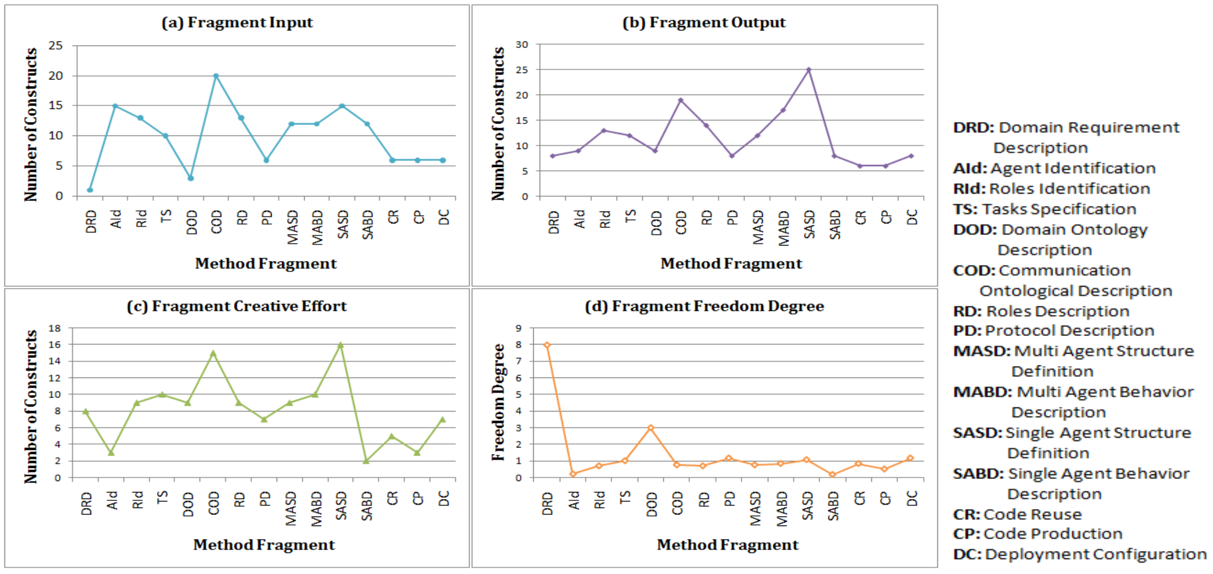


Figure 3: Evaluation of PASSI

4.2 Process Evaluation vs. Fragment Evaluation

As we saw in the previous subsection, the metrics that are proposed in this paper are of a good accuracy to evaluate fragment intrinsic characteristics but the correlated metric values are also of high interest for every single fragment.

A high value of FI is often accompanied by a high value of FCE and of FO, because the more constructs one has to design, and the more constructs are needed, the more is the effort spent in doing the design activity. This general rule is however not always respected, for instance for the *Agent Identification* fragment of PASSI. Although the FI and FO values are rather high, the FCE is rather low, particularly in the PASSI context, because of the fact that this portion of work only requires to group use cases in order to identify agents. All the information is already present in the previous fragment with which, in fact, there is a tight dependency also shown by the value of FI.

As the figures show, the metrics give also information on the global method characteristics. Some singularity points can be observed or comparative values of metrics can only be explained by a global view of the process. Let us illustrate this with some examples. The first one is about the singularities that were already mentioned on some ADELFE fragments that have very low values compared to their neighbourhood. The second one is for FI and FFD values of the *Domain Ontological Description* fragment in PASSI (cf. Fig.3.(a)(d)). As it can be seen, they are offbeat with the rest of the curve; this is because this fragment is not well positioned in the PASSI design process. This can be seen above all from the FFD where we expected to have a decreasing trend for the reasons said before. This result confirms what we thought about the position of this fragment and validate the accuracy of the proposed metrics. A last example is observed on the FFD curve for ADELFE (cf. Fig.2.(d)). As expected, the FFD flatten out at a low level during the requirement analysis phase (ended by the *UI Prototyping* fragment). However the FFD has a slight rise during the *Agent Identification* fragment and the *Agent Definition* fragment. Actually, the *Agent Identification* fragment aims at finding what agents will be considered in the system and the *Agent Definition* fragment aims to define the behaviour: skills, aptitudes, an interaction language, a world representation, etc. for every agent previously identified. In these fragments, designers have a high responsibility in the choice they make. As the figure shows, they have high freedom degree with little constructs already defined to guide them or to constraint them. All these examples find explanation in the relative values of the fragments of a same process rather than in the individual value of each one.

4.3 Discussion

A first difficulty occurring while comparing several methodologies is the lack of normalization. ADELFE is defined with 17 fragments while PASSI only contains 15 ones. And of course, they do not address the same phases with the same granularity. For this reason, we were obliged to normalize the curves in order to present ADELFE and PASSI with comparable values (cf. Fig.4). This was done by aligning fragments according to analysis, design and implementation phases. This alignment may also be needed on the Y axis that is the number of constructs. Obviously, the more constructs the metamodel contains, the higher the values will be in the metrics (except for FFD which is a ratio, naturally). This factor, which can be called the granularity of the metamodel, may happen when methods are using Model-Driven Development. In that case, metamodels are very big with plenty of details needed to tackle with accuracy required by model transformation algorithms. For this reason, it is important to normalize also the granularity with which a process is described and therefore the level of abstraction. None of the metrics takes the presence/adoption of Computer-Aided Software Engineering (CASE) tools into account. However, tools for guiding the engineer during the design are an advantage that should be evaluated by metrics. We will discuss about that in the following section.

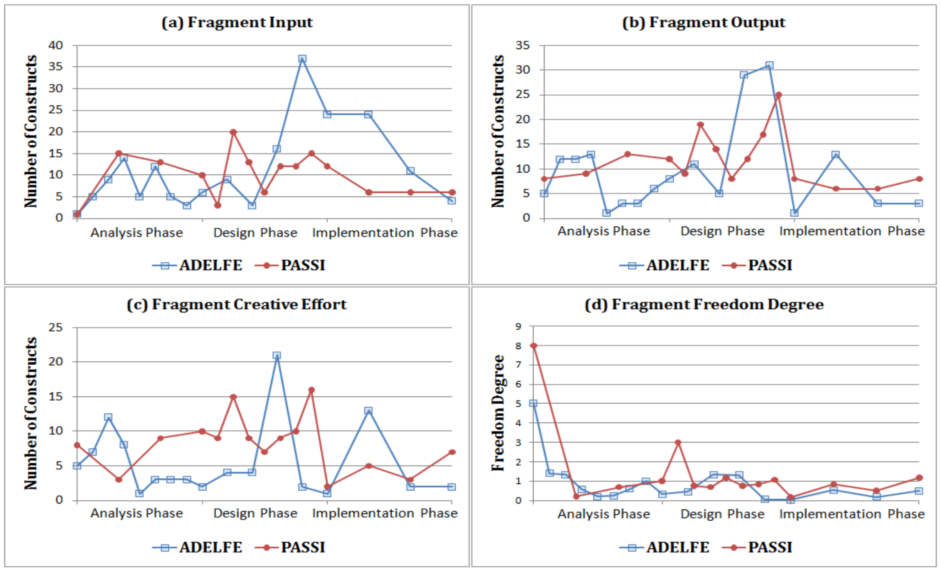


Figure 4: Comparison of ADELFE and PASSI

5. RELATED WORKS

Recently several AOM have been proposed. Thus far, however, software designers have not embraced any single methodology. In order to develop better solutions, designers need to understand advantages and limitations of existing methodologies. Therefore some works focus their efforts on the analysis of methodologies.

Cernuzzi's approach uses goal-question-metric to determine what factors are important to measure for comparing methods [4]. A qualitative analysis followed by a quantitative rating is proposed. Sturm and Shehory [22] develop a catalog of criteria for feature-based analysis of AOSE methodologies. *Agent-Based Characteristics* and *Software-Engineering Criteria* are differenciated. Their set of criteria is suitable to show the drawbacks of methodologies and therefore gives suggestions for further development. The above described approach compares the methodologies by a screening of the criteria. Dam and Winikoff [9] divided their found criteria into four dimensions to examine: (1) concepts and properties, (2) notations and modeling, (3) process, and (4) pragmatics. During the methodology evaluation, the correctness of a notation and the referenced characteristics is difficult to judge. The survey approach proposed here successfully reflects it. Moreover, based on both software engineering process principles and agent characteristics, Lin et al. [15] approach determines whether criteria have been met by the method and provides answers as statements for comparison from questions at the detailed level concerning logical relationships among these criteria.

The main lack of these approaches is that they only evaluate methodologies and do not take into account the portions which compose them. Currently, there is no tool that implements and simplifies the evaluation of the entire method process and their fragments.

6. CONCLUSIONS AND FUTURE WORKS

Despite all papers related to this topic there is no general and commonly adopted evaluation process of method. There is a fundamental need to have evaluation process in order to get a measurement of comparing completed activities of a method. In this paper, four objective dynamic metrics have been defined. These metrics enable analysing and comparing methods process and their fragment. Based on MAS metamodel, for each method fragment, particular numbers of constructs from the modelling actions performed on them are measured. Each measure enables to show some specificities of the fragments or some particularities of the method process. In order to illustrated the metrics, quantitative results of the agent-oriented method evaluation have been presented. This example usage illustrated how to derive method process features from the method fragment metamodels.

Future improvements to the presented metrics may result from an examination whether it is useful to consider different kinds of actions in workflow activity: (i) GUI action which is an activity performed by the designer using a GUI; (ii) automated action which is an activity performed by the tool to create a new constructs e.g. a model transformation; (iii) user action which is an activity not supported by a tool such as using a blackboard.

Currently, these metrics are basically used in a work of designed processes from self-combining method fragments. They enable the designed process evaluation at two levels: the evaluation of similar fragments and the evaluation of different processes.

7. REFERENCES

[1] F. Bergenti, M.P. Gleizes, and F. Zambonelli. *Methodologies And Software Engineering For Agent Systems: The Agent-oriented Software Engineering Handbook*. Kluwer Academic Pub, 2004.

[2] Carole Bernon, Marie-Pierre Gleizes, Gauthier Picard, and Pierre Glize. The Adelfe Methodology for an Intranet System Design. In *International Bi-Conferenystems (AOIS-2002) at CAice Workshop on Agent-Oriented Information SSE'02 (AOIS - SSE),*

Toronto, Ontario, Canada, 27-28 may 2002, page (on line), http://ceur-ws.org, May 2002. CEUR Workshop Proceedings.

[3] S. Brinkkemper, R.J. Welke, and K. Lyytinen. *Method Engineering: Principles of Method Construction and Tool Support*. Springer, 1996.

[4] Luca Cernuzzi, Gustavo Rossi, and La Plata. On the evaluation of agent oriented modeling methods. In *In Proceedings of Agent Oriented Methodology Workshop*, pages 21–30, 2002.

[5] M. Cossentino, S. Gaglio, A. Garro, and V. Seidita. Method fragments for agent design methodologies: from standardisation to research. *International Journal of Agent-Oriented Software Engineering (IJAOSE)*, 1(1):91–121, 2007.

[6] M. Cossentino and V. Seidita. Passi2 - going towards maturity of the passi process. *Technical Report ICAR-CNR*, (09-02), 2009.

[7] M. Cossentino and V. Seidita. Metamodeling: Representing and modeling system knowledge in design processes. Technical Report 11-02, Technical Report ICAR-CNR, 29 July 2011.

[8] Massimo Cossentino, Marie-Pierre Gleizes, Ambra Molesini, and Andrea Omicini. Process Engineering and AOSE (regular paper). In Marie-Pierre Gleizes and Jorge Gomez-Sanz, editors, *Workshop on Agent Oriented Software Engineering (AOSE), TORONTO - Canada, 10/05/2010-11/05/2010*, number 6038 in LNCS, pages 180–190, http://www.springerlink.com, 2011. Springer.

[9] Khanh Dam and Michael Winikoff. Comparing agent-oriented methodologies. In Paolo Giorgini, Brian Henderson-Sellers, and Michael Winikoff, editors, *Agent-Oriented Information Systems*, volume 3030 of *Lecture Notes in Computer Science*, pages 78–93. Springer Berlin / Heidelberg, 2004.

[10] Iván García-Magariño, Massimo Cossentino, and Valeria Seidita. A metrics suite for evaluating agent-oriented architectures. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 912–919, New York, NY, USA, 2010. ACM.

[11] Paolo Giorgini, Manuel Kolp, John Mylopoulos, and Jaelson Castro. Tropos: A requirements-driven methodology for agent-oriented software. chapter II, pages 20–45.

[12] AF Harmsen, S. Brinkkemper, and H. Oei. Situational method engineering for information system projects. In *Methods and Associated Tools for the Information Systems Life Cycle, Proceedings of the IFIP WG8. 1 Working Conference CRISí94*, pages 169–194, 1994.

[13] IEEE Foundation for Intelligent Physical Agents. *Design Process Documentation Template, Document number XC00097A-Experimental*, 2011.

[14] INGENIAS. Home page. `http://grasia.fdi.ucm.es/ingenias/metamodel/`.

[15] Chia-En Lin, Krishna M. Kavi, Frederick T. Sheldon, Kris M. Daley, and Robert K. Abercrombie. A methodology to evaluate agent oriented software engineering techniques. In *Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, HICSS 07, page 60, Washington, DC, USA, 2007. IEEE Computer Society.

[16] Mirko Morandini, Frédéric Migeon, Marie-Pierre Gleizes, Christine Maurel, Loris Penserini, and Anna Perini. A goal-oriented approach for modelling self-organising mas. In Huib Aldewereld, Virginia Dignum, and Gauthier Picard, editors, *Engineering Societies in the Agents World X*, volume 5881 of *Lecture Notes in Computer Science*, pages 33–48. Springer Berlin / Heidelberg, 2009.

[17] Object Management Group. *Meta Object Facility (MOF) Specification. http://doc.omg.org/formal/02-04-03*, 2003.

[18] OMG. Object Management Group. Software & Software Process Engineering Metamodel. version 2.0. Document number: formal/2008-04-01. 2008, 2008.

[19] J. Ralyté. Towards situational methods for information systems development: engineering reusable method chunks. *Procs. 13th Int. Conf. on Information Systems Development. Advances in Theory, Practice and Education*, pages 271–282, 2004.

[20] Sylvain Rougemaille, Jean-Paul Arcangeli, Marie-Pierre Gleizes, and Frédéric Migeon. ADELFE Design, AMAS-ML in Action. In *International Workshop on Engineering Societies in the Agents World (ESAW), Saint-Etienne, 24/09/2008-26/09/2008*, page (electronic medium), http://www.springerlink.com/, 2008. Springer-Verlag.

[21] V. Seidita, M. Cossentino, and S. Gaglio. A repository of fragments for agent systems design. *Proc. Of the Workshop on Objects and Agents (WOA06)*, 2006.

[22] Arnon Sturm and Onn Shehory. A framework for evaluating agent-oriented methodologies. In *Proc. of the Int. Bi-Conference Workshop on Agent-Oriented Information Systems, AOIS 2003, volume 3030 of LNCS*, pages 94–109, 2003.

[23] Quynh-Nhu N. Tran and Graham C. Low. *Comparison of Ten Agent-Oriented Methodologies*, chapter XII, pages 341–367. 2005.

Comma: A Commitment-Based Business Modeling Methodology and its Empirical Evaluation

Pankaj R. Telang and Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206
{prtelang,singh}@ncsu.edu

ABSTRACT

We introduce Comma, a methodology for developing cross-organizational business models. Comma gives prime position to patterns of business relationships understood in terms of commitments. In this manner, it contrasts with traditional operational approaches such as RosettaNet that are commonly used in industry.

We report the results of a developer study comparing Comma with a methodology recommended by the RosettaNet Consortium. Ours is one of the only evaluations of an agent-oriented methodology that (1) involves developers other than the proposing researchers and (2) compares against a traditional nonagent approach.

We found that Comma yields improved model quality, a greater focus in relative effort on the more important aspects of modeling, and a general reduction in total time despite yielding more comprehensive models. Certain anomalies in effort expended point toward the need for improved tooling.

Categories and Subject Descriptors

H.1.0 [**Information Systems**]: Models and Principles—*General*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Design, Experimentation

Keywords

Commitments, Business modeling, Methodology

1. INTRODUCTION

Real-world organizations seldom operate in isolation. To stay competitive, organizations develop deep expertise in core business functions, and outsource the rest to business partners. This results in a network of organizations with complex business relationships. Existing approaches for business modeling are of two broad types. The low-level approaches use concepts such as message ordering and control flow, and yield highly rigid models. The high-level approaches use concepts such as goals and values, and cannot be easily operationalized. Recently, researchers have begun to use social commitments for business modeling, e.g., [4], since they lead to flexible yet operationalizable models.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

We introduce Comma, a novel commitment-based business modeling methodology, which builds on a recent business metamodel [19]. Unlike traditional approaches, Comma gives prominence to patterns of business relationships. The motivation for developing abstractions such as commitments is that they would facilitate the engineering of superior solutions by helping build richer models of interaction. This is the main claim that we investigate here along with associated claims of ease of use and efficiency.

Two shortcomings of previous approaches are that, first, they do not adequately describe how to put concepts such as commitments into modeling practice, especially for the benefit of practitioners who are not multiagent systems specialists. And, second, previous approaches have not empirically evaluated their benefits in a controlled study, involving participants other than the authors. The same shortcomings, especially the second, might be said to apply on AOSE research broadly.

Contributions and Organization

The main contributions of this paper are the Comma methodology and a developer study comparatively evaluating it with respect to RosettaNet [14], a well-known traditional approach for cross-organizational processes. Our results confirm the relative effectiveness of Comma for the quality of modeling cross-organizational processes, and some benefits in ease of modeling and time expended. Further, the results yield insights for future improvements.

The rest of the paper is organized as follows. Section 2 describes the necessary background. Section 3 describes the Comma methodology. Section 4 outlines the design of the study, and Section 5 describes the study results. Section 6 discusses related work, and Section 7 concludes the paper with a discussion of future directions.

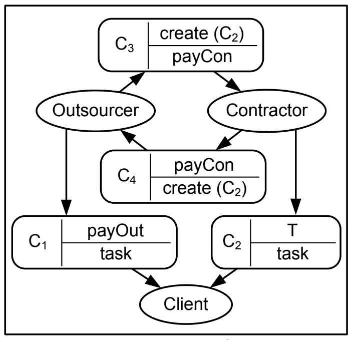
2. BACKGROUND

RosettaNet, a consortium of over 500 organizations, is a leading industry effort that develops standards for Business-to-Business integration that support business transactions worth billions of dollars. In RosettaNet, a Partner Interface Process (PIP) specifies a two-party interaction for a specific business intent. The PIPs are organized in a two-level hierarchy of *cluster* and *segment*. For example, Request Purchase Order PIP 3A4 is from Cluster 3 (Order Management) and Segment A (Quote and Order Entry). Using 3A4, a buyer sends a purchase order to a seller. Most PIPs define a two-party interaction involving a request and a response message. A modeler prepares a list of the necessary PIPs as the *RosettaNet model* of a business scenario. Next the modeler designs what we term *RosettaNet MSCs*: message sequence charts (MSCs) whose messages are derived from the PIPs.

We now describe some relevant concepts from Telang and Singh's [19] business metamodel. An *agent* models a real-world organization. The agent can play one or more roles in a business relationship. A *role* abstracts over the agents, and specifies, in a templatic form, the commitments that an agent adopting the role must participate in. A *task* is a business activity that an agent performs.

A *commitment* C(DEBTOR, CREDITOR, antecedent, consequent) means that the DEBTOR commits to the CREDITOR to bring about the consequent if the antecedent holds. The antecedent and the consequent are logical expressions over the tasks. When the antecedent of a commitment holds, the commitment detaches, and the debtor becomes unconditionally committed to the creditor to bring about the consequent. Regardless of the antecedent, if the debtor brings about the consequent, the commitment is satisfied [15]. For example, C = C(BUYER, SELLER, goods, pay) means that the buyer commits to the seller to paying if the seller ships the goods. C detaches if the seller ships the goods, and satisfies if the buyer pays regardless of when the seller ships the goods. Singh [16] explains commitments further.

Telang and Singh [19] define several business (modeling) patterns, of which our study used *commercial transaction*, and *outsourcing* patterns. We briefly describe the *outsourcing pattern*, and refer the reader to [19] for further details.



C_1	C(OUTSOURCER, CLIENT, payOut, task)
C_2	C(CONTRACTOR, CLIENT, $\top$, task)
C_3	C(OUTSOURCER, CONTRACTOR, create(C2), payCon)
C_4	C(CONTRACTOR, OUTSOURCER, payCon, create(C2))

Figure 1: Outsourcing business pattern [19].

Figure 1 shows the *outsourcing pattern* in Telang and Singh's notation. An oval represents a role; the label in the oval is the role name. A rounded rectangle represents a commitment. The rectangle shows the commitment name in the left-hand side, and in the right-hand side it shows the antecedent on the top and the consequent on the bottom. Two directed edges connect a commitment to roles: from the debtor to the commitment and from the commitment to the creditor. In the outsourcing pattern, an outsourcer delegates a task to a subcontractor. Here, C_1 is the original commitment from the outsourcer to a client to execute a task if the client pays the outsourcer (payOut). C_2 is the outsourced commitment from the contractor to the client to execute the same task. The antecedent of C_2 is true ($\top$), which means that it is unconditional. C_3 and C_4 are the commitments in which the outsourcer and the contractor commit to pay (payCon) and to create C_2, respectively.

3. COMMA

For each business pattern, such as those proposed by Telang and Singh [19], we develop a set of generalized (templatic) message sequence charts that operationalize that pattern.

Figure 2 shows the MSCs for the outsourcing pattern using UML 2.0 sequence diagram [11] operators OPT(ion) and ALT(ernative).

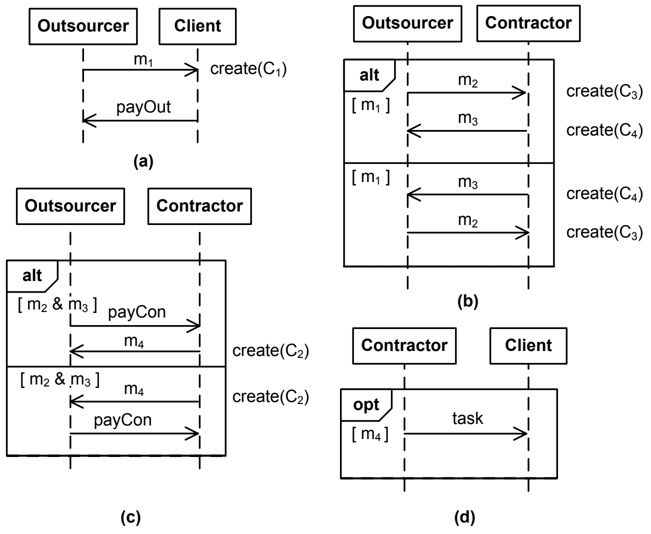


Figure 2: Message sequence charts for outsourcing.

We go beyond UML in labeling each message with its meaning. A message labeled with a proposition, usually part of the antecedent or consequent of some commitment, simply brings about that proposition. A message labeled m_i for some i means an operation on some commitment (such as its creation), which we annotate on the side. In Figure 2(a), the outsourcer sends m_1 to the client, which creates commitment C_1. The client sends payOut to the outsourcer upon receiving m_1, which detaches C_1 since it is C_1's antecedent. In Figure 2(b), after receiving m_1, the outsourcer sends m_2 to the contractor, and after receiving m_2 the contractor sends m_3 to the outsourcer. Alternatively, the contractor first sends m_3 to the outsourcer, and after receiving m_3, the outsourcer sends m_2 to the contractor. m_2 creates C_3 and m_3 creates C_4. In Figure 2(c), after m_2 and m_3 are exchanged, the outsourcer sends payCon to the contractor and the contractor sends m_4 to the outsourcer in either order. Now payCon satisfies C_3 and detaches C_4; and, m_4 creates C_2 and satisfies C_4. In Figure 2(d), after m_4 is exchanged, the contractor sends task (message) to the client. This satisfies C_1 and C_2 since task is their consequent. As part of creating a model, a modeler substitutes the message labels m_i with domain-specific terms.

The Comma methodology begins from an informally described real-life cross-organizational scenario and produces formal business and operational models. Table 1 summarizes Comma.

Step 1 A *subscenario* is a fragment of the given scenario. From the given scenario description, extract subscenarios such that each match a pattern from the Comma pattern library.

Step 2 For each subscenario, identify its roles. A subscenario usually describes participants using a combination of generic terms (e.g., Company, Partner, and Organization) and specific names (e.g., FedEx). This step involves creating roles based on business function (e.g., Shipper) that remove any ambiguity, such as if Partner and Organization refer to the same entity.

Step 3 For each subscenario, identify business tasks (e.g., goods and payment) that a role executes. A scenario typically specifies the tasks as actions executed by the participants.

Step 4 From the Comma pattern library, introduce into the business model a pattern corresponding to each subscenario. Rename the pattern characters with the roles from Step 2, and introduce the tasks from Step 3 as the antecedents and con-

Table 1: Comma methodology steps.

Step	Description	Input	Output
1	Extract subscenarios corresponding to Comma patterns	Real-life cross-organizational scenario	Subscenarios
2	Identify roles from each subscenario	Subscenario	Roles
3	Identify business tasks from each subscenario	Subscenario	Tasks
4	Introduce a Comma pattern for each subscenario	Comma pattern, subscenario, roles, tasks	Business model
5	Introduce MSCs for each Comma pattern	Comma pattern MSCs, subscenario, roles, tasks	Operational model

sequents of the appropriate commitments. The patterns compose naturally when the same roles are referenced by more than one pattern.

Step 5 For each Comma pattern, introduce its MSC into the operational model. Rename the roles and messages in the MSCs to align them with those determined in Steps 2 and 3. Customize the MSCs to capture any subscenario-specific operational details, such as additional messages, guards, and loops.

4. DESIGN OF THE STUDY

Our study used an initial scenario based on real-life cross-organizational business processes, inspired by the Oracle Quote-To-Cash (QTC) process [12, 19], and two modifications of the scenario.

$\mathcal{S}_i$, the initial scenario, involves MedEq, a company that sells medical equipment. MedEq designs the equipment in house, and out-sources manufacturing to two contract manufacturers, FlexMan and SoleMan, and shipping to two shippers, FedUp and UpFed. To purchase the equipment, a customer submits its requirements to MedEq. MedEq analyzes the requirements, and creates a proposal containing the equipment details, and a quoted price. The customer may accept the proposal or negotiate for a better price. There can be up to two iterations between MedEq and the customer before they either agree upon the price, or abort the transaction. If MedEq and a customer reach an agreement, the customer proceeds to placing an order and specifying the equipment, shipping address, contact information, and payment information. Upon receiving the order, MedEq validates the order. MedEq accepts the order if it is valid and rejects it otherwise. MedEq maintains warehouses in which it stocks the equipment. In case the ordered equipment is in stock, MedEq requests a shipper to ship the equipment to the customer. MedEq pays the shipping charges to the shipper.

If the equipment necessary to fulfill an order is not in stock, MedEq places a stock replenishment order with a contract manufacturer. The contract manufacturer employs a shipper to ship the equipment to MedEq's warehouse. MedEq pays the contract manufacturer for the equipment. Once the equipment is in stock, MedEq fulfills the customer's order.

$\mathcal{S}_f$, the first modification, adds a new participant, a value-added reseller, MedRes. MedRes sells, installs, and supports (i.e., services) medical equipment. The customer now places its order with MedRes, who orders the equipment from MedEq and provides the installation and support itself. The customer pays MedRes, and MedRes pays MedEq. MedRes supports the equipment as needed. The rest of the scenario remains unchanged.

$\mathcal{S}_s$, the second modification, removes the contract manufacturers SoleMan and FlexMan from the original scenario. The rest of the scenario is unchanged.

4.1 Study Solution

Figure 3 shows the solution Comma model for the initial scenario, $\mathcal{S}_i$. For brevity, we present only the final Comma model and omit the outputs of the intermediate methodology steps. The model is composed from the commercial transaction and the outsourcing patterns. For example, the commercial transaction pattern captures MedEq (Company) and the customer agreeing to exchange medical equipment for certain price. The model commitments C_1 and C_2 correspond to this pattern: in C_1, the customer commits to paying

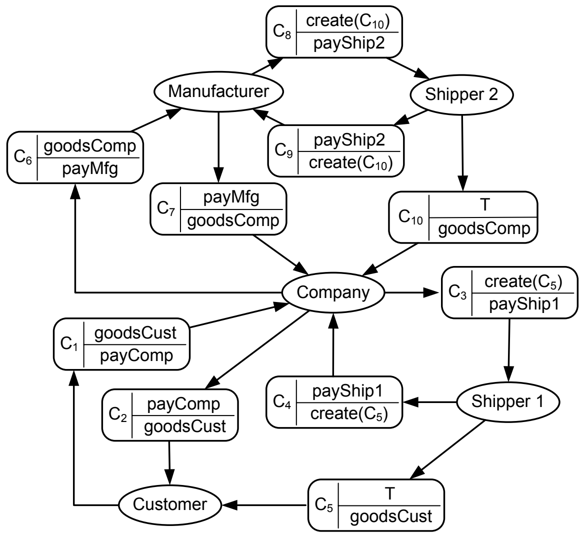


Figure 3: Comma model for $\mathcal{S}_i$.

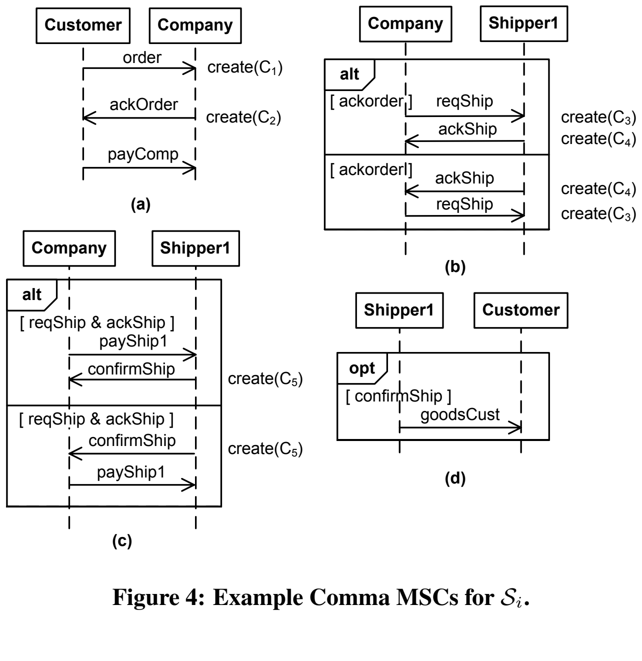


Figure 4: Example Comma MSCs for $\mathcal{S}_i$.

the company (payComp) if the company provides the equipment (goodsCust), and in C_2, the company commits to providing the equipment if the customer pays. The outsourcing pattern models MedEq employing a shipper (Shipper 1) to ship the medical equipment to the customer. The model commitments C_2, C_3, C_4, and C_5 correspond to this pattern: C_2 is the original commitment, C_5 is the outsourced commitment, and C_3 and C_4 are the commitments in which the company and the shipper commit to paying and to creating C_5, respectively. Figure 4 shows four of the ten MSCs for the initial scenario, $\mathcal{S}_i$, developed using Comma. These MSCs correspond to MedEq outsourcing the shipping to a shipper. We omit further description of these MSCs since Section 3 describes the outsourcing MSCs in detail.

Table 2: RosettaNet model PIPs for $\mathcal{S}_i$.

PIP	Name (shortened)	Subscenario
3A1	Request quote	Customer, MedEq negotiate
3A4	Purchase order	Customer orders from MedEq
3B12	Request shipping	MedEq ships to Customer
3C3	Notify of invoice	Shipper invoices MedEq, MedEq invoices customer, shipper invoices manufacturer, manufacturer invoices MedEq
3C4	Reject invoice	MedEq, customer, or manufacturer reject invoice
3C6	Remittance advice	MedEq pays the shipper, customer pays MedEq, manufacturer pays shipper, MedEq pays manufacturer
7B5	Manufacturing order	MedEq orders from manufacturer
3B12	Request shipping	Manufacturer ships to MedEq

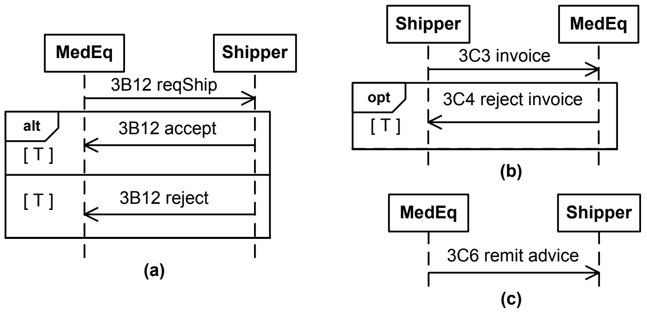


Figure 5: Example RosettaNet MSCs for $\mathcal{S}_i$.

Table 2 shows the RosettaNet model PIPs for the initial scenario, $\mathcal{S}_i$. For example, the customer uses PIP 3A1 to request a quote from MedEq. Figure 5 shows three of the thirteen MSCs for the initial scenario, $\mathcal{S}_i$, developed using RosettaNet. Figure 5(a) is the MSC for PIP 3B12 in which MedEq requests the shipper to ship the equipment to the customer. The shipper either accepts or rejects the request. The shipper invoices MedEq using PIP 3C3 in Figure 5(b). MedEq may reject the invoice using PIP 3C4. In Figure 5(c), MedEq notifies the shipper of remittance advice using PIP 3C6.

4.2 Study Mechanics and Threat Mitigation

We conducted a developer study with 34 subjects (graduate computer science students). Three exercises, corresponding to the three scenarios, $\mathcal{S}_i$, $\mathcal{S}_f$, and $\mathcal{S}_s$, comprised the study. The study used a *between-subject* experimental design [9]. For each exercise, the study divided the subjects into two groups who applied different methodologies to model the same scenario. We carefully designed the study to mitigate the well-known threats [9] to its validity.

To mitigate the threat of skill differences between the participants, prior to the exercises, we surveyed the study subjects to gather information on their educational background, and experience in process modeling and software engineering. We then divided the participants into two groups, *A* and *B*, of approximately equal skill levels. The first exercise compared groups *A* and *B*, and the subsequent exercises split and merged the same groups. For the first exercise, the subjects in groups *A* and *B* developed a model and MSCs for $\mathcal{S}_i$ using RosettaNet and Comma, respectively.

For the second and third exercises, a primary threat was the learning effect, because after the first exercise, subjects would be familiar with the methodology they used. To mitigate this threat, we divided each group into two subgroups of equal size and combined a subgroup from each group to form new groups *A'B'* and *A"B"*. A secondary threat was variance in the initial models developed by different subjects and their lack of familiarity with models developed by others. To mitigate this threat, we developed $\mathcal{C}$ and $\mathcal{R}$, respectively, Comma and RosettaNet model and MSCs for the initial scenario $\mathcal{S}_i$.

In the second exercise, group *A'B'* began from $\mathcal{C}$ and applied Comma, and group *A"B"* began from $\mathcal{R}$ and applied RosettaNet, both to account for $\mathcal{S}_f$.

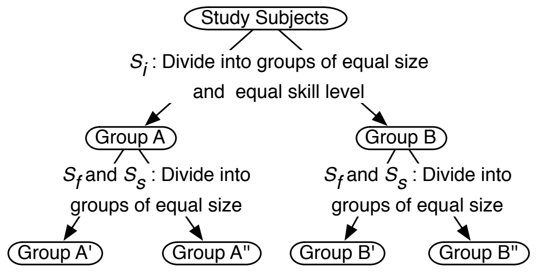

Study Subjects
S_i : Divide into groups of equal size and equal skill level
Group A
Group B
S_f and S_s : Divide into groups of equal size
S_f and S_s : Divide into groups of equal size
Group A'
Group A"
Group B'
Group B"

Figure 6: Our approach of grouping the subjects.

In the third exercise, we swapped the two groups. Group *A'B'* reviewed $\mathcal{R}$ and applied RosettaNet, and group *A"B"* reviewed $\mathcal{C}$ and applied Comma, both to account for $\mathcal{S}_s$.

Figure 6 summarizes how the study divided the subjects into groups, and Table 3 summarizes the exercises.

The subjects self-reported the time and difficulty for each methodology in a *work log*. To mitigate the threat of a subject forgetting to report relevant information, we required each subject to submit his or her work log three days a week, regardless of the effort they spent in that period.

4.3 Dependent Variables

This section describes the dependent variables of the study that we use to compare Comma and RosettaNet.

Quality of the models, assessed by experts, using the measures of Table 4. (A higher value is better for each.)

Difficulty in completing a methodology step as (subjectively) reported by a subject. Difficulty ranges over extremely easy, easy, neutral, difficult, and extremely difficult. Subjects reported the difficulty in a work log; we calculate the percentage of responses for each difficulty level. In most reports, we combine best two as *easy* and the worst two as *difficult*.

Table 3: Study exercises.

Exercise	Group A		Group B	
	Group A'	Group A"	Group B'	Group B"
1	Develop RosettaNet model and MSCs for $\mathcal{S}_i$		Develop Comma model and MSCs for $\mathcal{S}_i$	
2	Modify $\mathcal{C}$ to model $\mathcal{S}_f$	Modify $\mathcal{R}$ to model $\mathcal{S}_f$	Modify $\mathcal{C}$ to model $\mathcal{S}_f$	Modify $\mathcal{R}$ to model $\mathcal{S}_f$
3	Modify $\mathcal{R}$ to model $\mathcal{S}_s$	Modify $\mathcal{C}$ to model $\mathcal{S}_s$	Modify $\mathcal{R}$ to model $\mathcal{S}_s$	Modify $\mathcal{C}$ to model $\mathcal{S}_s$

Table 4: Quality measures, as judged by experts.

Measure	Captures a methodology's
Model Coverage. Percentage of models that fully cover the problem scenario	Completeness in modeling a scenario
Model Precision. Percentage of models that include no aspects unrelated to the problem scenario	Effectiveness in avoiding bloated models
MSC Structure. Percentage of MSCs with correct and complete guards	Soundness: fewer errors in outcomes
MSC Flexibility. Average number of ALT(ernative) blocks per MSC	Support for participants' flexibility
MSC Abstraction. Percentage of MSCs that use a role, not an agent, name	Support for reusability of models

Time taken to complete a methodology step as reported by a subject: a continuous variable in the unit of hours. Subjects reported the time they spent in a work log; we summed up the time for each subject.

5. STUDY RESULTS

This section describes the key findings from the study.

5.1 Quality

Figures 7 and 9 show the quality measurements of the two methodologies from the initial exercise $\mathcal{S}_i$.

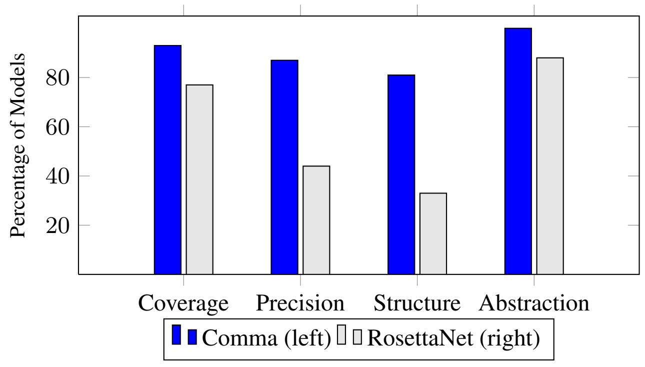


Figure 7: Quality of the models of $\mathcal{S}_i$.

Observation 1*:* As Figure 7 shows, both model coverage and model precision are superior for Comma (93% and 87%, respectively) than for RosettaNet (77% and 44%, respectively).

Observation 1 suggests that Comma is more effective than RosettaNet in creating complete and precise models. We credit this to the systematic nature of Comma and the fact that it focuses attention on the relevant commitments and MSCs. On the contrary, the RosettaNet models tend to contain several superfluous PIPs.

Observation 2*:* As Figure 7 shows, the percentage of models in which MSCs do not miss any necessary guards is higher for Comma (81%) than for RosettaNet (33%).

Since RosettaNet focuses on individual interactions in the form of PIPs, a modeler often loses an overall perspective on the scenario. The modeler develops an MSC for each PIP, but fails to relate the MSCs to each other via appropriate guards. In contrast, Comma forces a modeler to think in terms of the commitment life cycle. For example, a message that satisfies a commitment should be preceded by a message that creates the commitment.

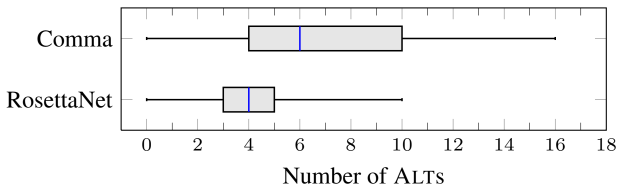


Figure 9: Flexibility of the MSCs produced for $\mathcal{S}_i$.

Observation 3*:* As Figure 9 shows, Comma MSCs use a higher median number of ALTs per model (six) than RosettaNet MSCs (four).

RosettaNet tends to lead to rigid MSCs, i.e., those with only a few alternative paths. The MSCs included with Comma patterns promote flexibility, which is inherent in the commitment-based approach. As a telling example, almost all subjects developed RosettaNet MSCs in which the Customer pays MedEq strictly after MedEq ships the ordered equipment. In contrast, many subjects developed Comma MSCs in which the Customer may pay MedEq either before or after MedEq ships the ordered equipment, a situation that has been discussed since the earliest works on commitment protocols [21].

Observation 4*:* The percentage of models in which MSCs use a role name instead of a participant name is higher for Comma (100%) than for RosettaNet (88%).

Observation 4 supports the idea that Comma emphasizes role abstraction and more naturally yields reusable MSCs.

Since the second and the third exercises began from the models that we provided, the resulting models are of higher quality, and without perceptible difference between the two methodologies. Therefore, we present quality results only for the first exercise.

5.2 Difficulty

Figure 8 shows the percentage of work log responses corresponding to each difficulty level for the three exercises.

Observation 5*:* In $\mathcal{S}_i$, the percentage of easy responses is smaller for RosettaNet (21.6%) than for Comma (27.5%), and the percentage of difficult responses is higher for RosettaNet (28.3%) than for Comma (23.7%).

Observation 5 suggests that Comma modeling is relatively easier as compared to RosettaNet modeling.

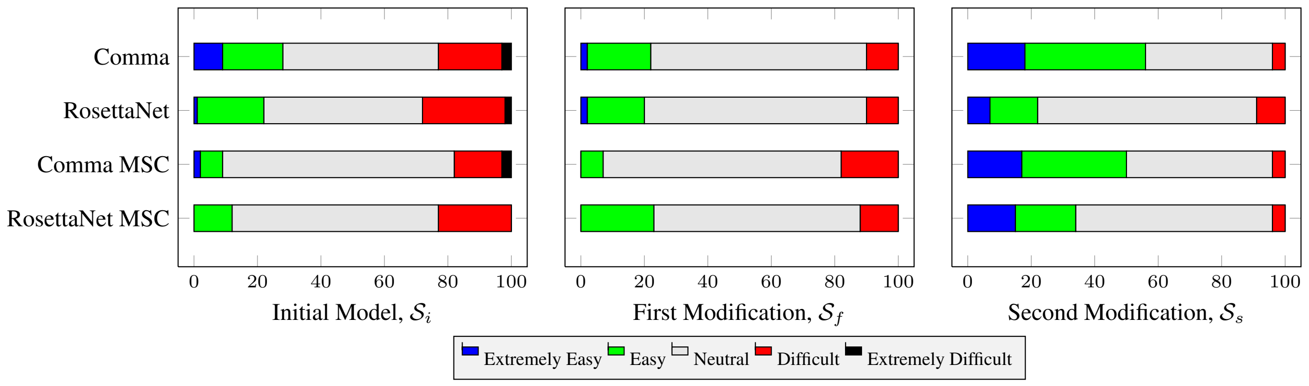


Figure 8: Difficulty of modeling, as percentage of responses by the subjects.

To identify the underlying cause of the extreme difficulty reports about Comma modeling, we analyzed the reported difficulty for each step. The analysis revealed that Comma Step 4, composing patterns to create a model, significantly contributes to the difficulty. This finding indicates the need for simplifying Step 4.

Observation 6*:* In $\mathcal{S}_i$, the percentage of difficult responses in developing MSCs using Comma (18.3%) is smaller than using RosettaNet (23.0%). However, the percentage of extremely difficult responses to developing MSCs using Comma (3.3%) is larger than using RosettaNet (0%).

Observation 6 is mixed. Although Comma appears to have been easier than RosettaNet overall, the number of subjects who found Comma extremely difficult was greater than the corresponding number for RosettaNet. This emphasizes the need for simplifying Comma Step 5, developing MSCs. A modeling tool, already under development, can assist a modeler by creating a base MSC model using the pattern MSCs.

Observation 7*:* Comma modeling has 0% extremely difficult responses, and 9.9% somewhat difficult responses in $\mathcal{S}_f$, as compared to 2.8% extremely difficult responses, and 20.9% percent somewhat difficult responses in $\mathcal{S}_i$.

We explain Observation 7 based on two factors. First, some of the subjects gained experience modeling using Comma in the initial exercise. Second, the subjects started the first modification $\mathcal{S}_f$ from a solution that we provided.

Relative to $\mathcal{S}_i$ and $\mathcal{S}_f$, $\mathcal{S}_s$ has increased responses with lower difficulty levels. This is partially due to the learning that the subjects gained from the first two exercises, and partially since $\mathcal{S}_s$ was a relatively easy exercise.

Observation 8*:* In $\mathcal{S}_s$, the percentages of easy responses for modifying the Comma model (56.2%) and MSCs (50%) are higher than for modifying the RosettaNet model (22.1%) and MSCs (33.3%).

Observation 8 suggests that with some experience, Comma becomes simpler than RosettaNet.

5.3 Time

Figure 10 shows boxplots of the time taken by the subjects to develop Comma and RosettaNet models and MSCs in the three exercises. Throughout, we remove each outlier: a point that is greater than the third quartile or smaller than the first quartile by 1.5 times the interquartile range—i.e., the difference between the third and first quartiles.

Observation 9*:* In $\mathcal{S}_i$, the median time to develop a model is smaller for Comma (6.7 hours) than for RosettaNet (10 hours).

Observation 9 suggests that Comma is more efficient than RosettaNet for creating a business model.

Observation 10*:* In $\mathcal{S}_i$, the median time to develop MSCs is somewhat greater for Comma (6 hours) than for RosettaNet (5.5 hours).

Although Comma appears less efficient than RosettaNet, as Section 5.1 shows, the MSCs produced from Comma are of higher quality than those produced from RosettaNet.

Observation 11*:* In $\mathcal{S}_i$, the spreads of the times for developing the model and MSCs are smaller for Comma than for RosettaNet.

Observation 11 indicates that Comma is more predictable than RosettaNet in terms of development effort.

Observation 12*:* Using Comma, the median modeling time for the first modification $\mathcal{S}_f$ (6.6 hours) is about the same as that for the initial exercise $\mathcal{S}_i$ (6.7 hours).

Observation 12 is surprising to us. We expected the Comma modeling time for $\mathcal{S}_f$ to be smaller than for $\mathcal{S}_i$. We attribute this result to a couple of key factors. First, the subjects needed time to comprehend the solutions we provided. Second, the subjects followed the same steps for modifying the model as the steps they followed for creating the model in the initial exercise. Comma should be improved to guide modelers in modifying existing business models.

Observation 13*:* In $\mathcal{S}_f$, the median modeling time is higher for Comma (6.6 hours) than for RosettaNet (4 hours).

Observation 13 conflicts with Observation 9 from the initial exercise $\mathcal{S}_i$. A primary reason for this result is the difference in the nature of the artifacts involved. A RosettaNet model is expressed as a textual list of PIPs, modifying which is easy. A Comma model is expressed as a graph of business relationships, modifying which is time consuming. Indeed, since we did not provide a Comma modeling tool, subjects expended considerable effort in developing the graphical models using drawing tools such as Visio.

Observation 14*:* In $\mathcal{S}_f$, the median time to modify MSCs is lower for Comma (1 hour) than for RosettaNet (2.3 hours).

Observation 14 suggests that the Comma methodology is more efficient as compared to the RosettaNet methodology for developing MSCs. Note that this result is an improvement over Observation 10 from the initial exercise $\mathcal{S}_i$ in favor of Comma, indicating the benefit of learning.

Observation 15*:* In $\mathcal{S}_s$, the median times to modify the Comma model (2.75 hours) and MSCs (0.75 hours) are slightly smaller than the median times to modify the RosettaNet model (3 hours) and MSCs (1 hour), respectively.

Observation 15 suggests that Comma is slightly more efficient

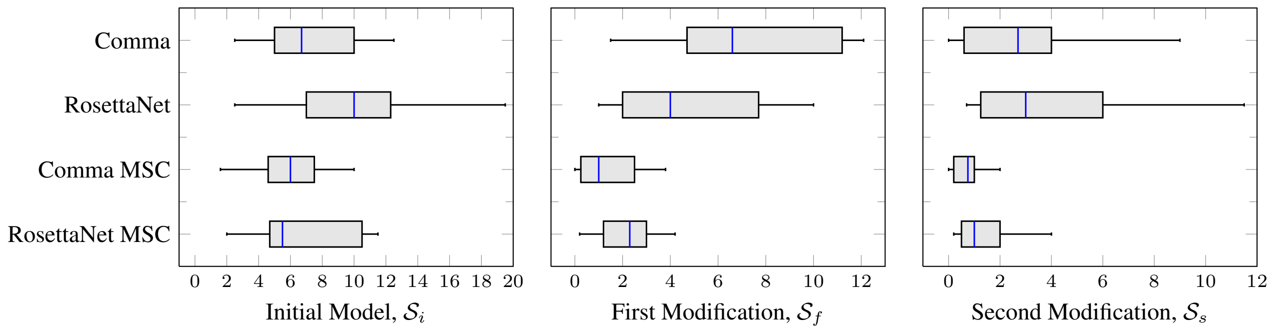


Figure 10: Time in hours expended in creating models, as reported by subjects.

Table 5: Hypothesis testing for model and MSC development times.

ID	Time for Exercise	Comma Mean (μ_c)	RosettaNet Mean (μ_r)	Alternative Hypothesis	Null Hypothesis $[\mu_c = \mu_r]$ p-value	Accepted at p-value of 5%?
H_1	$\mathcal{S}_i$-Model	7.19	10.05	$\mu_c < \mu_r$	0.046	×
H_2	$\mathcal{S}_i$-MSC	6.22	6.73	$\mu_c < \mu_r$	0.610	✓
H_3	$\mathcal{S}_f$-Model	7.59	4.84	$\mu_c > \mu_r$	0.026	×
H_4	$\mathcal{S}_f$-MSC	1.42	2.26	$\mu_c < \mu_r$	0.062	✓
H_5	$\mathcal{S}_s$-Model	2.77	3.74	$\mu_c < \mu_r$	0.290	✓
H_6	$\mathcal{S}_s$-MSC	0.70	1.29	$\mu_c < \mu_r$	0.053	✓

than RosettaNet for modifying models. This agrees with Observation 9 from the initial exercise $\mathcal{S}_i$.

Observation 16*:* In $\mathcal{S}_s$, the spreads of times taken in modifying the model and MSCs are smaller for Comma than for RosettaNet.

Observation 16 agrees with Observation 11, and reconfirms that Comma is more predictable than RosettaNet.

The above observations are from the descriptive statistics summarized by the box plots. We now present the results of formal hypothesis testing that checks if the difference between the timings of the two methodologies is statistically significant. Table 5 summarizes the hypotheses and the outcome of the *independent samples t-test* for each of them. H_1, H_3, and H_5 test the statistical significance of the difference between the modeling time of the two methodologies in $\mathcal{S}_i$, $\mathcal{S}_f$, and $\mathcal{S}_s$, respectively. H_2, H_4, and H_6 test the statistical significance of the difference between the MSC development time of the two methodologies in $\mathcal{S}_i$, $\mathcal{S}_f$, and $\mathcal{S}_s$, respectively. In H_1, the alternative hypothesis is $\mu_p < \mu_r$, that is, the mean time to develop the Comma model μ_p is less than the mean time to develop the RosettaNet model μ_r. The corresponding null hypothesis is $\mu_p = \mu_r$, that is, the mean time to develop the Comma model is the same as the mean time to develop the RosettaNet model. The t-test rejects the null hypothesis with *p value* of 0.046 at the 0.05 level of significance. This confirms that Comma is more efficient than RosettaNet in $\mathcal{S}_i$, which agrees with Observation 9.

The t-test rejects the null hypothesis in H_3. This indicates that RosettaNet is more efficient than Comma in the first modification $\mathcal{S}_f$. We discuss the reasons behind this result in Observation 13.

Since the t-test accepts H_2, H_4, H_5, and H_6, we conclude that the time differences for (1) modeling in $\mathcal{S}_s$ and (2) developing MSCs in all exercises is not statistically significant.

6. RELATED WORK

Researchers have proposed several agent-oriented software development methodologies [5, 13, 3, 20]. Many of these methodologies focus on modeling a multiagent system that is under the control of a single organization. In contrast, Comma models cross-organizational relationships. In Comma, a high-level model based on commitments captures the social relationship among agents (the organizations that are business partners). Unlike Comma, many of the current AOSE methodologies lack an appropriate abstraction for modeling social relationship between the agents.

Tropos [2] resembles Comma in terms of employing high-level concepts. A key difference between the two is how they model social relationships: Tropos employs goal and other dependencies whereas Comma employs commitments. Unlike dependencies, commitments are flexible as they can be manipulated. Commitments reflect the autonomy of the partners since each debtor adopts its commitments through its autonomous actions (communications).

Amoeba [6] employs commitment protocols for process modeling. Amoeba and Comma share the same underlying notion of commitments. In contrast to Comma, which is a methodology for business relationship modeling, Amoeba is a methodology for lower-level interaction modeling, and seeks to specify the protocols whose composition corresponds to the given business process.

Telang and Singh [18] approach RosettaNet from the opposite end to the present paper. They abstract out business modeling patterns from RosettaNet PIPs, in essence by identifying the commitments of the business partners involved that are implicitly understood in each PIP. That is, Telang and Singh discuss how to create and apply patterns that could be included in the Comma library. They use the commitment life cycle as a basis for verifying process specifications.

Mazouzi et al. [10] model agent interaction protocols using Agent UML (AUML), and subsequently translate them into Colored Petri Nets (CPN) to verify low-level properties such as liveness. In contrast, in Comma, a modeler first develops a high-level business model, which provides the correctness properties at a business level [19]. Starting from a business model, the modeler develops agent interaction MSCs. Comma employs model-checking to verify if the MSCs satisfy the business model [19].

Spanoudakis et al. [17] and Garcia-Magarino et al. [7] describe an application of Model-Driven Engineering (MDE) for AOSE. MDE can significantly improve the efficiency of Comma. A modeler can transform a Comma business model into an operational model, such as MSC, using automated model transformation.

Hofreiter et al. [8] describe UMM, UN/CEFACT's Modeling Methodology, a methodology to model inter-organizational business processes as global choreographies. Unlike Comma, UMM fails to capture the high-level business relationships between the process participants. Instead it focuses on the low-level message exchanges, and thus leads to rigid models.

7. CONCLUSIONS AND DIRECTIONS

We introduced Comma, a novel commitment-based methodology for business modeling. We carried out a substantial empirical evaluation of the effectiveness of Comma. We note in passing that such evaluations are not yet common in AOSE, though they are quite prevalent in the broader software engineering community.

Let us summarize the lessons we learned. Our study confirmed the benefits in quality that we expected from Comma because of its foundation in commitments. Specifically, Comma does better on every quality measure: model coverage and precision, and MSC structure (guards), flexibility, and abstraction. The study demonstrated gains in ease of use from Comma in producing models but yielded mixed results with respect to MSCs. Comma yields a superior MSC product, but with a slightly greater difficulty. We expect to see benefits from improving the tooling and training materials supporting Comma. The time spent shows an improvement for Comma though with anomalies. Here too we conjecture that improved tooling and training will prove crucial.

Some important future directions follow naturally from this research. First, on the theoretical side, we are considering expanding Comma to account for a richer variety of norms, e.g., in the spirit of Aldewereld et al. [1], than just commitments. Second, on the practical side, enhanced tooling is an obvious theme. A natural extension would be to support MDE using Comma, as remarked above. Further, we will enhance Comma so it provides guidance for situations where a model must be modified to accommodate evolving requirements.

Third, on the empirical side, we will conduct additional developer studies. Specifically, although our study design mitigated many important threats to validity that can arise in a comparative study, it did not consider important challenges to business interoperation in practice, such as dealing with a legacy system. We conjecture that increasing the complexity of a scenario will tilt the balance further in favor of commitment-based approaches: we defer such evaluations to future research. Further, a threat to validity of any empirical evaluation is whether the subjects correspond closely to the target population (industry practitioners, in our case) in their expertise, experience, and motivation. In their broadest scope, such problems are not readily amenable to comparative research studies, but we plan to explore simplified versions of them.

Acknowledgments

We thank Amit Chopra, Anup Kalia, Emerson Murphy-Hill, Scott Gerard and the anonymous reviewers for their helpful comments.

8. REFERENCES

[1] H. Aldewereld, S. Álvarez-Napagao, F. Dignum, and J. Vázquez-Salceda. Making norms concrete. *Proc. AAMAS*, pp. 807–814, 2010.

[2] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *JAAMAS*, 8(3):203–236, 2004.

[3] C. Cheong and M. P. Winikoff. Hermes: Designing flexible and robust agent interactions. V. Dignum, ed., *Handbook of Research on MAS*, ch. 5, pp. 105–139, 2009.

[4] A. K. Chopra, F. Dalpiaz, P. Giorgini, and J. Mylopoulos. Modeling and reasoning about service-oriented applications via goals and commitments. *Proc. CAiSE*, pp. 417–421, 2010.

[5] S. A. Deloach, M. F. Wood, and C. H. Sparkman. Multiagent systems engineering. *Int'l J. Soft. Engg. Know. Engg.*, 11(3):231–258, 2001.

[6] N. Desai, A. K. Chopra, and M. P. Singh. Amoeba: A methodology for modeling and evolution of cross-organizational business processes. *ACM TOSEM*, 19(2):6:1–6:45, Oct. 2009.

[7] I. García-Magariño, J. J. Gómez-Sanz, and R. Fuentes-Fernández. Model transformations for improving multi-agent systems development in INGENIAS. *Proc. AOSE 2009*, *LNCS* 6038, pp. 51–65, 2011.

[8] B. Hofreiter, C. Huemer, P. Liegl, R. Schuster, and M. Zapletal. UN/CEFACT's Modeling Methodology (UMM): A UML profile for B2B e-commerce. *2nd Int'l Wkshp. Best Pract. UML (ER)*, 2006, pp. 19–31.

[9] N. Juristo and A. M. Moreno. *Basics of Software Engineering Experimentation*. Kluwer, 2001.

[10] H. Mazouzi, A. E. F. Seghrouchni, and S. Haddad. Open protocol design for complex interactions in multi-agent systems. *Proc. AAMAS*, pp. 517–526, 2002.

[11] Object Management Group, *UML 2.0 Superstructure Specification*, Oct. 2004.

[12] Oracle. Automating the Quote-to-Cash process. 2009, http://www.oracle.com/us/industries/045546.pdf.

[13] L. Padgham and M. Winikoff. Prometheus: A practical agent-oriented methodology. *Agent-Oriented Methodologies*, ch. 5, pp. 107–135, 2005.

[14] RosettaNet, *Overview: Clusters, segments, and PIPs*, 2008, http://www.rosettanet.org.

[15] M. P. Singh. Semantical considerations on dialectical and practical commitments. *AAAI*, pp. 176–181, 2008.

[16] M. P. Singh. Commitments in multiagent systems. In F. Paglieri et al.(eds.) *The Goals of Cognition*, College Pubs., London, pp. 1–29, 2012. In press; available at http://www.csc.ncsu.edu/faculty/mpsingh/papers/drafts/Commitments-for-MAS.pdf

[17] N. Spanoudakis and P. Moraitis. Model-driven agents development with ASEME. *Proc. AOSE*, 2010.

[18] P. R. Telang and M. P. Singh. Abstracting and applying business modeling patterns from RosettaNet. *Proc. ICSOC*, pp. 426–440, 2010.

[19] P. R. Telang and M. P. Singh. Specifying and verifying cross-organizational business models. *IEEE Trans. Services Comput.*, 5, 2012. http://www.csc.ncsu.edu/faculty/mpsingh/papers/mas/TSC-11-business.pdf

[20] H. Weigand, V. Dignum, J.-J. C. Meyer, and F. Dignum. Specification by refinement and agreement: Designing agent interaction using landmarks and contracts. *Proc. ESAW*, *LNCS* 2577, pp. 257–269, 2002.

[21] P. Yolum and M. P. Singh. Commitment machines. *Proc. ATAL 2001*, *LNAI* 2333, pp. 235–247, 2002.

Revising Conflicting Intention Sets in BDI Agents[*]

Steven Shapiro
RMIT University
Melbourne, Australia
steven.shapiro@rmit.edu.au

Sebastian Sardina
RMIT University
Melbourne, Australia
sebastian.sardina@rmit.edu.au

John Thangarajah
RMIT University
Melbourne, Australia
john.thangarajah@rmit.edu.au

Lawrence Cavedon
RMIT University
Melbourne, Australia
lawrence.cavedon@rmit.edu.au

Lin Padgham
RMIT University
Melbourne, Australia
lin.padgham@rmit.edu.au

ABSTRACT

Autonomous agents typically have several goals they are pursuing simultaneously. Even if the goals themselves are not necessarily inconsistent, choices made about how to pursue each of these goals may well result in a set of intentions which are conflicting. A rational autonomous agent should be able to reason about and modify its set of intentions to take account of such issues. This paper presents the semantics of some preferences regarding modified sets of intentions. We look at the possibility of simply deleting some intention(s) but more importantly we also look at the possibility of modifying intentions, such that the goals will still be achieved but in a different way.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent Agents*

General Terms

Theory

Keywords

BDI, Logic-based approaches and methods, Formal models of agency, Modeling the dynamics of MAS

1. INTRODUCTION

BDI (Belief, Desire, Intention) systems (for an overview, see Bordini *et al.* [1]) are a popular approach to modelling and implementing agent systems. It is well accepted and reflected in the theoretical underpinnings of BDI systems (e.g., Rao and Georgeff [14]) that rational agents should not intend to pursue a goal that they believe is impossible to achieve. Similarly, we believe that an agent should not have a *set* of intentions, which taken together are unachievable. Such situations can readily arise as new goals are committed to, with new intentions instantiated, but without reasoning about how the independent intentions interact. Intention reconsideration—i.e., revisiting the commitments to planned activity held by an agent—is considered an important notion in BDI agent conceptual frameworks (e.g., Bratman [2] and Bratman *et al.* [3]). Intention revision is a central component of this process.

In this paper, we explore the semantics of the kinds of preferences that should guide the reasoning of an autonomous agent that is revising its intentions for some reason. Our focus is to provide a principled semantics for evaluating the different options for revising an existing intention set. The most straightforward way to revise an intention set is to drop one or more intentions. However, this will lead to lack of achievement of the associated goal(s). A preferred but more complex option is to modify one or more intentions (i.e., modify how we intend to achieve the associated goals) to obtain a non-conflicting intention set.

As an example, consider a purchasing agent with an intention to purchase a laptop and another intention to purchase a printer. If the agent is made aware that funds are insufficient for both purchases, then it may drop one intention such that the other is achievable. However, if it is possible to modify its current intentions, for example by changing its choice of laptop to be a cheaper model so that both intentions can be satisfied, this would be a preferred revision.

We consider three basic principles in defining our semantics: the first is *environmental tolerance*—we prefer a set of intentions which can succeed in more environments; the second is a *maximal cardinality* principle—keep as many top-level goals as possible; and the third is a *minimal modification* principle—if we must change the means chosen to achieve a goal (i.e., an intention), we prefer to change it as little as possible. These principles necessarily interact and we have to make particular choices as to which should dominate as we develop the semantics. However, minor modifications would allow the relationships to be changed.

Our long term goal is to incorporate rules into the execution engine of BDI agent programming languages to support principled intention revision. In doing this, we must choose between a quantitative framework—requiring costs associated with actions and rewards with goals, as well as perhaps probability distributions—and a qualitative one. The former would offer more flexibility, but requires the programmer to provide required quantities and measures. As typical BDI programming languages generally lack facilities for specifying such things as costs, rewards and probability distributions, we choose the qualitative approach. Consequently, we base our framework on the CAN family of languages [17] as this maps well to languages used in a number of BDI agent development platforms [1], but also has been extended to include advanced reasoning techniques, such as planning and reasoning about goals.

We stress that we are presenting here a semantics for how agents ought to revise their intentions, and not an implementation. However, in future work, we plan to develop an implementation that is both faithful to the semantics and computationally viable.

[*]We acknowledge the ARC Discovery grant DP1094627.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

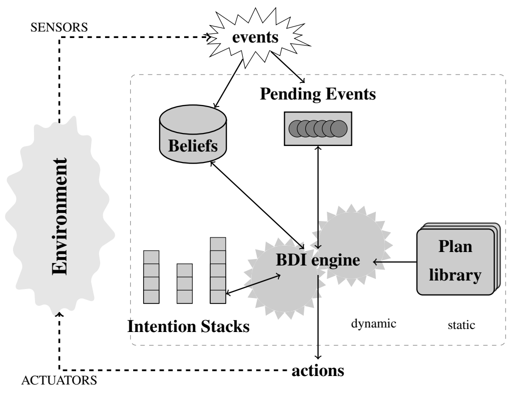


Figure 1: A typical BDI-style architecture [17].

In the following sections, we first present the basic constructs of the CAN language. We then provide the semantics for revising intention sets using only deletion, followed by the semantics when we allow modification of individual intentions. We finish with a discussion of related work, as well as some further issues to be addressed in the future.

2. AGENT PROGRAMMING IN THE CAN FRAMEWORK

BDI agent-oriented programming is based on formal computational models, such as the ones proposed by Cohen and Levesque [4] and Rao and Georgeff [14], and the philosophical work of Bratman [2] and Dennett [6], using mental attitudes, such as beliefs, desires, goals, plans and intentions. Practically, BDI agent systems enable *abstract plans* written by programmers to be combined and used in real-time, in a way that is both flexible and robust. The CAN (**C**onceptual **A**gent **N**otation) family of BDI languages [17] are AGENTSPEAK-like languages with a semantics capturing typical BDI systems (for an overview, see Bordini *et al.* [1]).

A typical BDI system (see Figure 1) responds to *events*[1]by selecting a plan from the system's know-how information for execution. The execution of a plan may, in turn, post new subgoal events to be achieved.

In the CAN language, there are three types of atoms: events (denoted by e), basic beliefs (denoted by b), and actions (denoted by *act*). A propositional language $\mathcal{L}_B$ is formed in the usual manner using the basic beliefs as atoms. We will use ψ, possibly with decorations, to denote a sentence of $\mathcal{L}_B$.

In CAN, a BDI agent is specified by an initial belief base $\mathcal{B}$, a plan library Π, and an action description library Λ. $\mathcal{B}$ is a model of the agent's initial beliefs about the world. It is a set of belief atoms that the agent holds to be true. We use $\mathcal{B} \models \psi$ to denote that the sentence ψ is true in belief base $\mathcal{B}$. This is defined in the usual manner.

An action description library Λ encodes the effects of primitive actions. It is a set of STRIPS-style operators of the form: $act : \psi \leftarrow \Phi^+; \Phi^-$, one for each action atom in the domain. Here, ψ is the precondition of the action, and is restricted to be a conjunction of belief literals. Φ^+ and Φ^- are sets of belief atoms, and correspond to STRIPS-style add and delete lists, respectively. Note that we assume all actions are deterministic.

[1]In CAN, there is no distinction between events and goals, and we sometimes refer to them as event-goals.

2.1 Plan Library

The *plan library*, Π, encodes the operational information of the domain via plan rules of the form $e : \psi \leftarrow P$, where e is an event, ψ is the context condition, and P is the plan-body program—*P is a reasonable strategy for resolving event e when condition ψ is believed to be true*. Plan-body programs are built from the following constructs, which we call the *user program language*:[2]

act	primitive action
$+b$, $-b$	add/delete a belief atom
$?\psi$	test for a condition
$!e$	post an event-goal
$P_1; P_2$	sequence

A program that only contains constructs from the user program language is called a *user program*.

In the *full program language*, there are two additional constructs used internally for defining the semantics of programs (i.e., they are not used in the programs in the plan library):

nil	the empty (terminating) program
$P \triangleright e : (\!| \psi_1 : P_1, \ldots, \psi_n : P_n |\!)$	attempt P to achieve e

In the last construct, P is a program, e is an event-goal, and:

$$(\!| \psi_1 : P_1, \ldots, \psi_n : P_n |\!)$$

is a set of alternative guarded plans relevant to e. The semantics for this construct is that an execution for P is attempted, and only if P fails, an alternate plan whose context condition is satisfied, say P_1, is selected for execution. In that case, the construct may transition to: $P_1 \triangleright e : (\!| \psi_2 : P_2, \ldots, \psi_n : P_n |\!)$.[3]

For example:[4]

$$!Buy(pc, shop) \triangleright Get(pc) : (\!| isOnline : !Buy(pc, web), \\ isOffline : !Buy(pc, mailOrder) |\!)$$

could be the intention of a purchasing agent to attempt to purchase a laptop from the shop, and if that fails to fall back to achieving the goal $Get(pc)$ either via the Web, if the agent is online, or by mail-order, if the agent is offline. There are standard ways to axiomatise a programming language such as the one presented here (see, e.g, Hennessy [10]), and we assume such an axiomatisation without presenting the details here. The language we use for this axiomatisation and for expressing properties of our framework is second-order, since we will be quantifying over functions, e.g., environments (which are defined below). We also assume we have an axiomatisation of finite sets, and omit the details here.

2.2 Intention Base

The *intention base*, Γ, of an agent contains the programs that the agent has already committed to for handling previously posted events. Formally, it is a set of programs in the full program language.[5] It is an element of the semantics of the CAN language, described below, but we present it separately since it is central to the framework described here; it is the intention base of an agent that is revised. We illustrate the intention base with an example.

[2]Note that the CAN language also contains a concurrency construct and allows variables in context conditions, but we omit those here for simplicity.

[3]Note that this is a variant of the CAN language, where $\triangleright$ and $e : (\!| \Delta |\!)$ are treated as separate constructs.

[4]As noted above, the version of CAN we consider here is propositional. However, we will sometimes use atoms with parentheses as syntactic sugar, e.g., $Buy(pc, shop)$.

[5]In CAN, an intention is actually a pair containing an identifier (e.g., a natural number) and a program, however we suppress the identifier for simplicity.

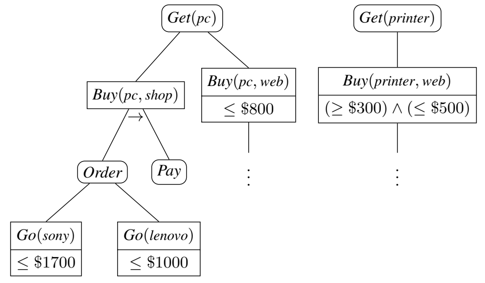


Figure 2: An example goal-plan hierarchy. Plans are represented with rectangles; goals are represented with rounded rectangles. Arcs with an arrow between them represent sequential composition. Arcs with no arrow represent alternate plans for their parent goal. The second annotation on plans states how much it will end up costing given the set of possible environments.

EXAMPLE 1. *Consider a purchasing agent with two intentions: one to achieve the goal of purchasing a laptop and another to achieve the goal of buying a printer. Figure 2 outlines the available plans in the library for achieving these goals. Assume* $\Gamma = \{I_1, I_2\}$*, where:*

$$I_1 = [(Go(sony) \rhd Order\colon (\!|\psi_1 : Go(lenovo)|\!)); !Pay] \rhd \\ Get(pc) : (\!|\psi_2 : Buy(pc, web)|\!);$$
$$I_2 = \ !Get(printer).$$

That is, the agent has already started addressing the task of buying a laptop and has chosen to buy a Sony. However, should it fail to buy a Sony, it may alternatively try to buy a Lenovo (if context condition ψ_1 holds, e.g., the Lenovo store is open) or even fall back to acquiring a laptop on the Web (if condition ψ_2 applies, e.g., it can access the Internet).

2.3 Semantics

As with most agent programming languages, the Plotkin-style operational semantics of CAN closely follows Rao and Georgeff's [15] abstract interpreter for rational agents, and is defined using a so-called BDI *agent configuration* C of the form $\langle \Pi, \Lambda, \mathcal{B}, \mathcal{A}, \Gamma \rangle$, where components $\Pi, \Lambda, \mathcal{B}$, and Γ (resp.) are the plan library, action library, belief base and intention base (resp.), as described above, and $\mathcal{A}$ is the sequence of actions executed so far. A *transition relation* $C \longrightarrow_{\mathcal{E}} C'$ on agent configurations is then used to state that a *single step* in executing agent configuration C yields configuration C' within an environment $\mathcal{E}$. We represent an environment as a function $\mathcal{E} : Confs \mapsto 2^{Events}$, where *Confs* is the set of all possible agent configurations and *Events* is the set of all possible external events in the domain, i.e., programs of the form: $+b$, $-b$, and $!e$. That is, an environment determines which external actions occur at each step in the execution of an agent system. We further require that environments are consistent in the sense that at most one of $+b$ and $-b$ are in $\mathcal{E}(C)$, for any belief atom b and configuration C. For example, given a configuration C and an environment $\mathcal{E}$, $\mathcal{E}(C)$ might contain a belief update that the price of Lenovo laptops have dropped 30% (e.g., due to a sale). Finally, a *BDI execution* E of an agent $C_0 = \langle \Pi, \Lambda_0, \mathcal{B}_0, \mathcal{A}_0, \Gamma_0 \rangle$ in an environment $\mathcal{E}$ is a, possibly infinite, sequence of agent configurations $C_0 \cdot C_1 \cdots$ such that $C_i \longrightarrow_{\mathcal{E}} C_{i+1}$, for all $i \geq 0$.

Given an intention $I \in \Gamma_0$ and an execution E (w.r.t. an environment $\mathcal{E}$), it is possible to determine whether the intention I has successfully executed (i.e., it has evolved to the empty program, *nil*), failed, or can continue to execute. In general, an intention fails if it becomes *blocked*, that is, it cannot execute further in the current configuration (for example, if an action does not meet its precondition or a test step is believed false), and it cannot be "recovered" via the default failure handling mechanism of re-trying different alternatives for active (sub)goals. We refer to Sardina and Padgham [17] for full details on the semantics of the language.

In this paper, we focus mainly on the last component of configuration $C = \langle \Pi, \Lambda, \mathcal{B}, \mathcal{A}, \Gamma \rangle$, namely, the intention base Γ. It will be notationally convenient to separate this component out from the rest of the configuration. We therefore define the *diminished configuration* C_- to be a configuration with the intention base argument omitted, i.e., $C_- = \langle \Pi, \Lambda, \mathcal{B}, \mathcal{A} \rangle$. Where confusion does not arise, we will simply refer to these as *configurations*. $\langle C_-, \Gamma \rangle$ will be used to denote the (full) configuration that has Γ as its last component, and whose other components are as in C_-.

3. REVISION BY DROPPING INTENTIONS

Our aim is to characterise which possible intention bases Γ^* qualify as *acceptable intention revisions* for a given intention base Γ that the agent deems to be in need of reconsideration. We specify a *semantics* for when a new intention base counts as an appropriate revision of the current intentions. In future work, we plan to investigate an implementation that could be used in actual BDI systems which would be faithful to this ideal. We are currently agnostic on when and why a given intention base ought to be revised. This could happen, for example, when a new belief or intention is added, or when the agent determines there may be a negative interaction or conflict between intentions. One way to achieve a similar result to revising intentions would be to simply execute the intentions and the ones that end up being blocked would be dropped automatically from the intention set. However, executing intentions could consume valuable resources that could be saved by revising the intention set at appropriate times. In this section, we focus on the revision of intention sets by simply *abandoning* one or more current intentions.

One of the dimensions we use for comparing intention sets consists of the environments in which the intention set can be successfully executed. However, which intention sets can be successfully executed in an environment depends on how smart the agent is about the choices it has to make during an execution, e.g., about which plans to select and intentions to execute at which times. A smart agent, e.g., one who can plan ahead to make the right decisions, might be able to execute reliably a given intention set. On the other hand, a dumb agent—e.g., one who does not plan ahead, but rather chooses an intention to execute according to a fixed rule such as round-robin—might not be able to execute successfully the same intention set reliably. So, the definition of a successful execution depends on the agent under consideration. In the interest of generality—so that our framework can apply to a variety of agents—we take the successful execution of a set of intentions Γ in a configuration C_- and an environment $\mathcal{E}$, $Success(\Gamma, C_-, \mathcal{E})$, to be a primitive of our framework, and we leave it undefined. We only make the assumption that the empty set of intentions always executes successfully, since there is nothing to execute.

AXIOM 1 (**SUCCESS**).

$$\forall C_-, \mathcal{E}.\, Success(\emptyset, C_-, \mathcal{E}).$$

Let S denote this axiom together with the axioms for finite sets.

For example, let us return to our purchasing agent described in Example 1. Suppose the agent's budget is \$1700 and that the set of possible environments are those described in Figure 2 with the second annotation in plans (e.g., buying a computer on the web will cost no more than \$800). In this case, the agent may be unable to *successfully* execute (both) its intentions I_1 and I_2 (in environments where buying a Sony and a printer costs over \$1700) and hence one purchase may eventually fail (e.g., at time of payment for the printer, if the computer is purchased for \$1500). The question then is: *can the agent revise* Γ *to a more "robust" set of intentions?*[6]

The various dimensions in qualitative frameworks, such as the one we have chosen, are often incomparable, and therefore, they cannot be readily combined. Consequently, decisions must be made as to which dimensions take precedence. In our framework, one dimension for comparison consists of the environments in which an intention set can be successfully executed (*environmental tolerance*), and another is the cardinality of the intention set.[7] In this paper, we give precedence to environmental tolerance. However, the framework could easily be adapted so that cardinality gets precedence. Having environmental tolerance dominate cardinality means the agent is *cautious* in that it would prefer to drop an intention to gain more certainty that the intention set will be successfully executed. Alternatively, giving precedence to cardinality would model a *bold* agent that believes things will "work out" (with respect to the environment), and prefers to maintain more of its intentions. Both can lead to counterintuitive results in particular examples, but this is inherent in qualitative frameworks that contain incomparable dimensions.

At a minimum, once the agent has decided to revise its intentions, the revised intention set should be executable in *some* environment. We prefer intention sets that are executable in more environments. Of these preferred sets, we choose the sets that have higher cardinality, i.e., retain more intentions from the original set.

We define the revision of an intention set in three stages. First, we define what it means for an intention base to be *maximal* in terms of environmental tolerance. Amongst the maximal bases, the revision *candidates* are those that have some chance of success. We then say that the revision sets are the largest candidate sets.

3.1 Definition of a Maximal Set

We say that a set of intentions Γ *dominates* another set Γ' in a configuration C_-, if the agent can successfully execute Γ in a superset of the environments in which it successfully executes Γ' in C_-. Formally:

DEFINITION 2 (**DOMINATES**).

$$\mathit{Dom}(\Gamma, \Gamma', C_-) \overset{\mathit{def}}{=} \forall \mathcal{E}.\, \mathit{Success}(\Gamma', C_-, \mathcal{E}) \supset \mathit{Success}(\Gamma, C_-, \mathcal{E}).$$

We then define the set of maximal options in terms of dominance. A set of intentions $\overline{\Gamma}$ is maximal with respect to a set Γ in configuration C_-, if $\overline{\Gamma}$ is a subset of Γ and it dominates any nonempty subset of Γ that dominates it. The empty intention base needs to be ruled out for comparison as it (trivially) dominates *every* intention set—the empty set of intentions always executes successfully. We want to allow the possibility of the agent abandoning all its intentions, if no subsets of its intentions have any chance of success, however, we also want this to be the only condition under which the agent drops all of its intentions.

DEFINITION 3 (**MAXIMAL**).

$$\mathit{Max}(\overline{\Gamma}, \Gamma, C_-) \overset{\mathit{def}}{=} \overline{\Gamma} \subseteq \Gamma \wedge [\forall \Gamma'.\Gamma' \subseteq \Gamma \wedge \Gamma' \neq \emptyset \supset (\mathit{Dom}(\Gamma', \overline{\Gamma}, C_-) \supset \mathit{Dom}(\overline{\Gamma}, \Gamma', C_-))].$$

Observe here that the empty set is indeed maximal, but other options can be maximal as well.

3.2 Definition of a Revision Set

For an intention base to be a revision candidate of Γ, it must be a maximal subset of Γ, but also *possible* in the sense that the intentions can be achieved in at least one environment. Note that the second constraint does not follow from maximality: if no nonempty subset of Γ succeeds, then every subset will be considered maximal. Formally, we define $\mathit{Cand}(\cdot,\cdot,\cdot)$ as follows:

DEFINITION 4 (**REVISION CANDIDATE**).

$$\mathit{Cand}(\overline{\Gamma}, \Gamma, C_-) \overset{\mathit{def}}{=} \mathit{Max}(\overline{\Gamma}, \Gamma, C_-) \wedge \mathit{Poss}(\overline{\Gamma}, C_-), \textit{where } \mathit{Poss}(\Gamma, C_-) \overset{\mathit{def}}{=} \exists \mathcal{E}.\, \mathit{Success}(\Gamma, C_-, \mathcal{E}).$$

Finally, an intention revision Γ^* of an intention base Γ is defined to be a largest candidate set:

DEFINITION 5 (**REVISION**).

$$\textsc{Rev}(\Gamma^*, \Gamma, C_-) \overset{\mathit{def}}{=} \mathit{Cand}(\Gamma^*, \Gamma, C_-) \wedge \forall \Gamma'.\, \mathit{Cand}(\Gamma', \Gamma, C_-) \supset |\Gamma'| \leq |\Gamma^*|.$$

In Example 1, there are two possible revisions: $\Gamma_1 = \{I_1\}$, and $\Gamma_2 = \{I_2\}$, i.e., the agent drops either one of its intentions. These sets dominate $\Gamma = \{I_1, I_2\}$ because they terminate in strictly more environments than Γ does (i.e., all environments where Sony + printer cost $>$ \$1700).

3.3 Properties

We now turn to some properties of these definitions.[8] Firstly, a (possibly empty) revision set always exists:

PROPOSITION 6.

$$S \models \forall \Gamma, C_-.\exists \Gamma^*.\, \textsc{Rev}(\Gamma^*, \Gamma, C_-).$$

Secondly, consider the (optimal) case in which it is possible to select a (nonempty) *fully robust* set of intentions, i.e., a set of intentions that can be fully executed in *every* environment. In that case, any revision will be fully robust:

PROPOSITION 7.

$$S \models \forall \Gamma, C_-.\, (\exists \Gamma'.\Gamma' \subseteq \Gamma \wedge \Gamma' \neq \emptyset \wedge \mathit{SuccessAlways}(\Gamma', C_-)) \supset (\forall \Gamma^*.\, \textsc{Rev}(\Gamma^*, \Gamma, C_-) \supset \mathit{SuccessAlways}(\Gamma^*, C_-)),$$
$$\textit{where } \mathit{SuccessAlways}(\Gamma, C_-) \overset{\mathit{def}}{=} \forall \mathcal{E}.\, \mathit{Success}(\Gamma, C_-, \mathcal{E}).$$

The empty set of intentions can always be achieved in any configuration. However, we want the agent to drop all its intentions *only* if there is no other choice, i.e., when none of the other subsets of Γ have any chance of success. Indeed, we can show that the revision of a set Γ is the empty set *iff* no nonempty subset of Γ is possible.

PROPOSITION 8.

$$S \models \forall \Gamma, C_-.[\forall \Gamma'.\Gamma' \subseteq \Gamma \wedge \Gamma' \neq \emptyset \supset \neg \mathit{Poss}(\Gamma', C_-)] \equiv [\forall \Gamma^*.\, \textsc{Rev}(\Gamma^*, \Gamma, C_-) \supset \Gamma^* = \emptyset].$$

[6]Note that in our example, the order of the selection of plans has already been fixed, and the order of execution of intentions is irrelevant, therefore the exact definition of *Success* is not important.

[7]Later on, we will introduce a minimal modification dimension.

[8]All propositions in this paper were verified with the PVS Verification System [12].

This proposition shows that given a set Γ that has no chance of success, although we cannot guarantee the success of a revision of Γ, any non-empty revision of Γ will at least have some chance of success. Furthermore, a revision of Γ will only be empty if no nonempty subset of Γ has any chance of success.

We close by noting that with the method for revision considered here, the only choice is to drop current intentions altogether. In the context of BDI agent systems, where goals can be achieved in multiple ways, one can envision more sophisticated accounts of revision so as to avoid resorting to such drastic decisions. We consider one such account in the next section.

4. REVISION BY MODIFICATION

In the previous section, we considered resolving conflicting intentions by dropping some. However, BDI agents often have various alternative plans for achieving goals. Instead of dropping an intention altogether, another option is to see if adopting other alternatives resolves the conflict. This option, which we call *intention modification* is preferable because it can lead to the achievement of more of the agent's goals.

Consider again our purchasing agent from Example 1 with the two intentions to buy a computer and a printer. Instead of dropping one of its intentions, the agent can consider modifying its intention to buy a computer by changing the method of achieving the event-goal *Order* to the plan $Go(lenovo)$, or indeed by changing the means to achieve the top-level event-goal $Get(pc)$ to the plan $Buy(pc, web)$. Either of these changes will result in the successful execution of both intentions, as the total cost in any possible environment will be below the available funds of \$1700.

The definition of revision by modifying intentions is more involved than the definition of revision by dropping intentions. First, we must define what we mean by modifying an intention. We modify an intention by selecting an alternate plan to achieve the intention or a subintention. This only makes sense for what we call *active goals* of the intention, which are the (sub)goals of the intention for which a plan has already been selected to achieve the goal. If P' is a modification of P, then we call P' an *alternate* of P. Then, we define the *alternate subset* relation which holds between two intention sets Γ' and Γ, if every element of Γ' is an alternate of an element of Γ. With this definition in hand, we define a *cardinal revision* of an intention set Γ in a similar fashion to the definition of revision in previous section, except we replace subset with alternate subset. This gives us a largest, maximal set in the space of alternate subsets of Γ, rather than in the space of subsets of Γ, as before. We define what it means to be a "least modification" of an intention, and generalise that definition to hold over sets of intentions, which we call *setwise closeness*. Finally, we define a revision of an intention set Γ^* to be a cardinal revision of Γ^* that is maximal with respect to setwise closeness. Although the formal framework in this section is more complicated than the previous one, conceptually the differences are simple: 1) the candidate sets for a revision of Γ are taken from the set of alternate subsets of Γ rather than the set of subsets of Γ; and 2) we add an extra dimension to compare the revision candidates for Γ, namely, minimal modification of the elements of Γ.

4.1 Definition of Alternate Subset

To define revision by modification, we must first define what we mean by a modification. We take a modification to be a different choice of a plan to achieve the intention or a subgoal of the intention. Formally, we say that P' is an *alternate* of a program P relative to a belief base $\mathcal{B}$, $P \rightsquigarrow_{\mathcal{B}} P'$ if P' can be obtained from P by changing the way some (sub)goal in P is to be achieved. We define this as a set of recursive axioms, with three cases depending on the structure of P.[9]

AXIOM 2 (**ALTERNATE OF A PROGRAM**).

$$(P_1; P_2) \rightsquigarrow_{\mathcal{B}} P' \equiv \exists P^*. P_1 \rightsquigarrow_{\mathcal{B}} P^* \wedge P' = (P^*; P_2);$$

$$\begin{aligned} &P_1 \rhd e\colon (\!|\Delta|\!) \rightsquigarrow_{\mathcal{B}} P' \equiv \\ &\quad (\exists P^*. P_1 \rightsquigarrow_{\mathcal{B}} P^* \wedge P' = (P^* \rhd e\colon (\!|\Delta|\!))) \vee \\ &\quad \exists \psi, P.\ \psi\colon P \in \Delta \wedge \mathcal{B} \models \psi \wedge P' = (P \rhd e\colon (\!|\Delta \setminus \{\psi\colon P\}|\!)); \end{aligned}$$

$$P_1 \rightsquigarrow_{\mathcal{B}} P_2 \equiv P_1 = P_2, \textit{otherwise}.$$

In other words, if the program is a sequence $P_1; P_2$, then we recursively look for alternates in P_1. We do not need to consider alternates in P_2, since while the agent may have started executing P_1, and therefore may have generated alternative plans to achieve its (sub)goals, this would not yet be the case for P_2. If the program is of the form $P_1 \rhd e\colon (\!|\Delta|\!)$, then we recursively look for alternates in P_1 (first disjunct), but we also allow other choices of programs in Δ to achieve e, provided their context conditions are satisfied by $\mathcal{B}$ (second disjunct). In this case, P_1 is dropped as a means to achieve e. Note that when P_1 is dropped, *all* subgoals of P_1 are automatically dropped as well. For all other program constructs, the only alternate is the program itself.

EXAMPLE 9. *In Example 1, the alternates to I_1 are I_1 itself ($\rightsquigarrow_{\mathcal{B}}$ is reflexive), and assuming ψ_1 and ψ_2 are satisfied:*

$$\begin{aligned} I_1' &= \big((Go(lenovo) \rhd Order\colon (\!| \ |\!)); !Pay\big) \rhd \\ &\qquad\qquad Get(pc)\colon (\!|\psi_2 : Buy(pc, web)|\!); \\ I_1'' &= Buy(pc, web) \rhd Get(pc)\colon (\!| \ |\!). \end{aligned}$$

The basic idea for revision by modification is that, given a set of intentions Γ to revise, we consider not just subsets of Γ, but also for each subset of Γ, we replace the elements of the subset by each of their alternates and consider the resulting sets as well. From these sets, we choose the ones with greatest environmental tolerance that are possible, and among those, the ones that are largest in size. From the remaining sets, we take the ones with the *least changes* to the elements of Γ to be the revisions of Γ. The aim in minimising the changes to the intentions is, as far as possible, to retain the work already done by the agent to achieve the associated goal.

Given the definition of an alternate of a program, we can now define the *alternate subset* relation ($\Gamma' \sqsubseteq_{\mathcal{B}} \Gamma$), which holds if every element of Γ' is an alternate of an element of Γ, and no two elements of Γ' are alternates of the same element of Γ:

DEFINITION 10 (**ALTERNATE SUBSET**).

$$\begin{aligned} &\Gamma' \sqsubseteq_{\mathcal{B}} \Gamma \overset{\textit{def}}{=} (\forall I' \in \Gamma'. \exists I \in \Gamma. I \rightsquigarrow_{\mathcal{B}} I') \wedge \\ &\quad \forall I_1', I_2' \in \Gamma'. \forall I \in \Gamma. I \rightsquigarrow_{\mathcal{B}} I_1' \wedge I \rightsquigarrow_{\mathcal{B}} I_2' \supset I_1' = I_2'. \end{aligned}$$

4.2 Definition of Cardinal Revision

In considering intention modifications, we only consider alternatives of *active goals* in each intention structure.[10] An active goal is the event-goal e in a program of the form $P \rhd e : (\!|\Delta|\!)$, where e is the active goal and Δ is a set of guarded plans.[11]

The *goal-program trace* of a program P is a finite sequence of pairs of the form $(e, (\!|\Delta|\!))$, consisting of an active goal e, and its alternate guarded plans $(\!|\Delta|\!)$, and is defined as follows. We treat this recursive definition as an axiom:

[9]We adopt the convention that unbound variables are universally quantified in the widest scope.

[10]All other goals are either completed, or not yet expanded so the alternatives are unknown.

[11]Note that this definition is different from the one given in [17], where the whole program was considered the active goal.

AXIOM 3 (**GOAL-PROGRAM TRACE**).

$$Trace(P) = \begin{cases} Trace(P_1) & \text{if } P = P_1; P_2 \\ (e, (\!|\Delta|\!)) \cdot Trace(P_1) & \text{if } P = P_1 \rhd e : (\!|\Delta|\!) \\ \epsilon & \text{otherwise,} \end{cases}$$

where the operator $\cdot$ is sequence concatenation and ϵ is the empty sequence. Note that we assume P has been evolved from a user program using the transition rules for CAN. For such programs, it can be shown that if P is of the form $P_1; P_2$ (case 1), none of the goals in P_2 are active yet. Therefore, we only have to consider the goal-program trace of P_1. For example, the goal-program trace of intention I_1 in Example 1 is:

$$(Get(pc), (\!|\psi_2 : Buy(pc, web)|\!)) \cdot (Order, (\!|\psi_1 : Go(lenovo)|\!)).$$

DEFINITION 11. *Let Σ denote the set of axioms consisting of Axioms 1–3, along with the axioms defining the language of* CAN, *and the axiomatisation of finite sets.*

Note that the framework in the previous section was independent of the agent language used, therefore we did not need the axioms for the language of CAN. However, revision by modification is framed in terms of the constructs of the CAN language. This is evident, for example, in Axiom 2. However, it would not be difficult to adapt the definitions to other agent languages.

We assume that each intention in the intention base has a different top-level goal. For $\Gamma \sqsubseteq_{\mathcal{B}} \Gamma^*$, this allows us to uniquely identify the subset of Γ^* whose alternate is Γ. The semantics of the CAN language ensures that an intention I in the intention base can only be of the form $!e$, or $P \rhd e : (\!|\Delta|\!)$. We formally define the *top-level goal* of an intention as follows:

DEFINITION 12 (**TOP-LEVEL GOAL**).

$$TLG(I) \overset{def}{=} e, \text{when } I = !e, \text{or } Trace(I) = (e, (\!|\Delta|\!)) \cdot \ldots$$

Note that the trace of $!e$ is ϵ, so the second case does not apply to intentions of the form $!e$. We say that a set of intentions is *distinct*, if the top-level goals of its members are all different:

DEFINITION 13 (**DISTINCT INTENTION SET**).

$$Distinct(\Gamma) \overset{def}{=} \forall I, I' \in \Gamma . I \neq I' \supset TLG(I) \neq TLG(I').$$

Next, we update the definitions *Max* and *Cand* from the previous section by substituting *subset* with *alternate subset* ($\sqsubseteq_{\mathcal{B}}$). Let C_{-}^{B} denote the belief base component of C_{-}. We define:

DEFINITION 14 (**MAXIMAL INTENTION SET**).

$$Max^{+}(\overline{\Gamma}, \Gamma, C_{-}) \overset{def}{=} \overline{\Gamma} \sqsubseteq_{C_{-}^{B}} \Gamma \wedge [\forall \Gamma' . \Gamma' \sqsubseteq_{C_{-}^{B}} \Gamma \wedge \Gamma' \neq \emptyset \supset (Dom(\Gamma', \overline{\Gamma}, C_{-}) \supset Dom(\overline{\Gamma}, \Gamma', C_{-}))].$$

DEFINITION 15 (**CANDIDATE INTENTION SET**).

$$Cand^{+}(\overline{\Gamma}, \Gamma, C_{-}) \overset{def}{=} Max^{+}(\overline{\Gamma}, \Gamma, C_{-}) \wedge Poss(\overline{\Gamma}, C_{-}).$$

In deciding which candidate sets are better than others, we now have two dimensions to consider. In the previous section, we chose the candidate sets with the largest cardinality. This amounts to preferring to keep top-level goals. As we will see below, for revision by modification, we will also prefer sets whose intentions have undergone the least degree of modification. For the moment, we use an intermediate predicate, REVCARD^{+} to capture the former preference. We say that Γ^* is a *cardinal revision* of Γ in C_{-}, if Γ^* is a candidate for revision and is no smaller than all other candidates.

DEFINITION 16 (**CARDINAL REVISION**).

$$\text{REVCARD}^{+}(\Gamma^*, \Gamma, C_{-}) \overset{def}{=} Cand^{+}(\Gamma^*, \Gamma, C_{-}) \wedge \forall \Gamma' . Cand^{+}(\Gamma', \Gamma, C_{-}) \supset |\Gamma'| \leq |\Gamma^*|.$$

As one would expect, a cardinal revision always exists:

PROPOSITION 17.

$$\Sigma \models \forall \Gamma, C_{-} . Distinct(\Gamma) \supset \exists \Gamma^* . \text{REVCARD}^{+}(\Gamma^*, \Gamma, C_{-}).$$

4.3 Definition of Revision by Modification

Now, we formally specify what it means to be a "least modification" of an intention. Given belief base $\mathcal{B}$, and programs P, P_1 and P_2 such that $P \rightsquigarrow_{\mathcal{B}} P_1$ and $P \rightsquigarrow_{\mathcal{B}} P_2$, we say that P_1 is *closer* than P_2 to P (i.e., $P_1 \preceq_P P_2$), if the following holds (as standard, $|\cdot|$ denotes the length of a sequence):

DEFINITION 18 (**CLOSER**).

$$P_1 \preceq_P P_2 \overset{def}{=} |Trace(P_1)| \geq |Trace(P_2)| \wedge (P_2 = P \supset P_1 = P).$$

In other words, P_1 is closer to P than P_2 if, basically, no more goals have been dropped in P_1 than in P_2 (with respect to P). If no goals have been dropped in either, then P_2 is closer to P than P_1, if P_2 is identical to P, but P_1 is not (because in P_1, an alternate plan has been chosen to achieve the last subgoal). We capture that case via the second conjunct. For example, using the intentions in Examples 1 and 9, it can be seen that $I_1' \preceq_{I_1} I_1'' \wedge I_1'' \not\preceq_{I_1} I_1'$, since $|Trace(I_1')| > |Trace(I_1'')|$; I_1' is strictly closer to I_1 than I_1''. To verify that $\preceq_P$ is intuitively correct, we show that the length of the *Trace* of an alternate P' of a program P is not greater than the length of the *Trace* of P itself.

PROPOSITION 19.

$$\Sigma \models \forall \mathcal{B}, P, P' . P \rightsquigarrow_{\mathcal{B}} P' \supset |Trace(P)| \geq |Trace(P')|.$$

This property holds because an alternate P' of P can have dropped goals but not added any.

For our definition of revision by modification, we must generalise $\preceq_P$ to apply to two alternate subsets of the set Γ of intentions to be revised. We only compare sets that consist of alternates of the same subset of Γ. If $\Gamma' \sqsubseteq_{\mathcal{B}} \Gamma$, then the elements of Γ' are each alternates of the elements of a subset of Γ, which we call the *core* of Γ' with respect to Γ and $\mathcal{B}$. Formally, given $\Gamma' \sqsubseteq_{\mathcal{B}} \Gamma$:

DEFINITION 20 (**CORE OF AN INTENTION SET**).

$$core(\mathcal{B}, \Gamma, \Gamma') \overset{def}{=} \{P \in \Gamma | \exists P' \in \Gamma' . P \rightsquigarrow_{\mathcal{B}} P'\}.$$

Note that if Γ is distinct, then Γ' is guaranteed to be the same size as $core(\mathcal{B}, \Gamma, \Gamma')$, for any $\mathcal{B}$.

For alternate subsets Γ_1 and Γ_2 of Γ that share the same core, we say that Γ_1 is *setwise closer* to Γ than Γ_2, if for every $I \in \Gamma$, I's alternate in Γ_1 is closer to I than its alternate in Γ_2:

DEFINITION 21 (**SETWISE CLOSER**).

```
$\Gamma_1 \preceq_{\mathcal{B},\Gamma} \Gamma_2 \overset{def}{=}$
  **if** $\Gamma_1 \sqsubseteq_{\mathcal{B}} \Gamma \wedge \Gamma_2 \sqsubseteq_{\mathcal{B}} \Gamma \wedge core(\mathcal{B}, \Gamma, \Gamma_1) = core(\mathcal{B}, \Gamma, \Gamma_2)$
  **then** $\forall I \in \Gamma, I_1 \in \Gamma_1, I_2 \in \Gamma_2 . I \rightsquigarrow_{\mathcal{B}} I_1 \wedge I \rightsquigarrow_{\mathcal{B}} I_2 \supset$
                $I_1 \preceq_I I_2$
  **else** $FALSE$,
```

where **if** A **then** B **else** $C \overset{def}{=} (A \supset B) \wedge (\neg A \supset C)$.

Finally, we say that Γ^* is a *modification revision* of Γ in C_-, if Γ^* is a cardinal revision and is maximal with respect to setwise closeness:

Definition 22 (**Modification Revision**).

$$\begin{aligned}&\textsc{Rev}^+(\Gamma^*, \Gamma, C_-) \overset{def}{=} \textsc{RevCard}^+(\Gamma^*, \Gamma, C_-) \wedge \\ &\quad \forall \Gamma'.\, \textsc{RevCard}^+(\Gamma', \Gamma, C_-) \supset \\ &\qquad (\Gamma' \preceq_{C^B_-, \Gamma} \Gamma^* \supset \Gamma^* \preceq_{C^B_-, \Gamma} \Gamma').\end{aligned}$$

4.4 Properties

We can show that (the appropriate reformulations of) the propositions in the previous section hold for $\textsc{Rev}^+$ as well.

A (possibly empty) modification revision set of a distinct set Γ always exists.

Proposition 23.

$$\Sigma \models \forall \Gamma, C_-.\, \mathit{Distinct}(\Gamma) \supset \exists \Gamma^*.\, \textsc{Rev}^+(\Gamma^*, \Gamma, C_-).$$

If there exists a fully robust subset of a distinct set Γ, then the modification revision set will be fully robust.

Proposition 24.

$$\begin{aligned}&\Sigma \models \forall \Gamma, C_-.\, \mathit{Distinct}(\Gamma) \wedge \\ &\quad (\exists \Gamma'. \Gamma' \sqsubseteq_{C^B_-} \Gamma \wedge \Gamma' \neq \emptyset \wedge \mathit{SuccessAlways}(\Gamma', C_-)) \supset \\ &\qquad (\forall \Gamma^*.\, \textsc{Rev}^+(\Gamma^*, \Gamma, C_-) \supset \mathit{SuccessAlways}(\Gamma^*, C_-)).\end{aligned}$$

The modification revision of a distinct set Γ is the empty set *iff* no nonempty subset of Γ is possible.

Proposition 25.

$$\begin{aligned}&\Sigma \models \forall \Gamma, C_-.\, \mathit{Distinct}(\Gamma) \supset \\ &\qquad [\forall \Gamma'. \Gamma' \sqsubseteq_{C^B_-} \Gamma \wedge \Gamma' \neq \emptyset \supset \neg \mathit{Poss}(\Gamma', C_-)] \equiv \\ &\qquad [\forall \Gamma^*.\, \textsc{Rev}^+(\Gamma^*, \Gamma, C_-) \supset \Gamma^* = \emptyset].\end{aligned}$$

Let us return to our purchasing agent example, as described in Examples 1 and 9, and in Figure 2. As there is \$1700 available, the (best) revision is obtained by dropping the goal of buying a Sony and adopting the goal of buying a Lenovo (for achieving event-goal *Order*), that is, the only modification revision is $\Gamma^* = \{I_1', I_2\}$. Observe that while intention base $\Gamma' = \{I_1'', I_2\}$ can be considered for revision, it involves higher-level modifications to intentions than Γ^* does. That is, let $\mathcal{B}$ be the agent's current belief base, then $I_1' \preceq_I I_1'' \wedge I_1'' \not\preceq_I I'$, and therefore $\Gamma^* \preceq_{\mathcal{B},\Gamma} \Gamma' \wedge \Gamma' \not\preceq_{\mathcal{B},\Gamma} \Gamma^*$, which means that Γ' does not qualify as an acceptable revision set. However, if the agent were to believe there was only \$1400 available, then intention I_1 would have to be modified higher up in its hierarchy of goals, and the only revision would indeed be Γ'. Observe that in both cases, though, dropping any of the intentions, as in the previous section, would not qualify as acceptable revisions: there is something less drastic that can be done.

5. RELATED WORK

The problem of revising intentions, especially within formal BDI frameworks, has received surprisingly little attention. Rao and Georgeff [16] extend their seminal BDI logic to resolve a specific paradox that arises when intentions are dropped but do not consider the general problem. Wobcke [19] considers the effects on intentions when beliefs are revised; he uses a version of *epistemic entrenchment* [7], effectively requiring a quantitative measure of the priority of each intention. To our knowledge, Wobcke was the first to apply ideas from belief revision [7] to the problem of revising intentions. However, his framework, and in particular his representation of plans, is restrictive.

More recently, Grant *et al.* [8] present a detailed investigation of the general problem of intention revision. They propose postulates for the revision of *BDI structures*; a BDI structure $\langle B, D, I, v, (c, C)\rangle$ is a representation of the current mental state of an agent (similar to what we call a configuration), where v computes the "value" in achieving a desire and c assigns a "cost" to performing actions in the domain C (which includes all actions the agent may intend) that are used to achieve goals. Each intention in I is represented as a plan instance or "recipe" $a \rightarrow g$, where a is an action to perform and g is the goal (i.e., proposition) that would be achieved by successful execution of a. Grant *et al.* are specifically interested in BDI structures that are "rational" (are not internally inconsistent) and that maximise "benefit" (the total value of its current goals minus the cost of the actions to achieve them).[12]

Grant *et al.* propose postulates for the various cases of adding and deleting a belief, desire or intention, and for the cases of modifying value and cost functions; each of these cases may result in a new BDI structure. Their main requirements for the resulting BDI structure are: (i) the rationality condition; (ii) that any change to the intention set is minimal (in that it overlaps as much as possible with the intention set in the original BDI structure); and (iii) that the candidate revisions are maximal in benefit.

While our aims have many similarities to those of Grant *et al.*, we have taken an orthogonal approach by considering intention revision within the rich CAN framework. In particular, this framework allows us to reason about complex plans or recipes to be used to achieve goals, and the environments in which they can be successfully performed, allowing us to revise to intention sets that are maximally likely to be successful. The rich plan representation, including subgoals, available in the CAN language also allows us to consider the possibility of revising to a new intention by considering alternative execution paths to achieve a goal or subgoal. In particular, use of the CAN framework better bridges the gap between abstract theoretical framework and the standard execution model of BDI agent systems [1, 13]. It also allows us to reason explicitly about revising to sets of maximally robust intentions, and impose our minimal modification condition.

Van der Hoek *et al.* [18] define a powerful framework for describing and reasoning about BDI mental states, and a corresponding account for revising intentions. Their ultimate goal is to incorporate the dynamics of intentions into a framework for reasoning about mental states; as such, they have similar aims to ours. Their formal framework is very rich; however, their approach to the actual intention revision is very algorithmic, which is an issue that Grant *et al.* [8] specifically attempt to address. Further, the model presented in Van der Hoek *et al.* does not address two of our main concerns: their intention-removal step does not seem to account for mutually conflicting intentions; and they do not enforce minimal modification of intentions.

Finally, Icard *et al.* [11] also consider how belief revision impacts intention revision; they also tightly intertwine the revision of beliefs and intention sets. They provide postulates for revising both sets of intentions and beliefs. However, this is under a simple model of plans, in particular, non-hierarchical plans; hence, they are not able to capture a notion of minimal plan modification such as the one presented here.

Most of the other frameworks discussed here examine the relationship between beliefs, goals and intentions, which is something

[12]In a follow-up paper, Grant *et al.* [9] extend this framework to the multi-agent case, to model a different problem, that of how agents use knowledge of each others' intentions to coordinate behaviour. Since this work is only tangentially related to ours, we will not discuss it further here.

which we do not address. The reason we do not address this issue is that although the interplay between these mental attitudes is theoretically interesting, we do not see it playing a role in currently implemented systems. For example, in the CAN framework, as well as in many implemented agent systems, goals are simply labels with no propositional meaning. Similarly, while beliefs do have propositional content in CAN, the language in which beliefs are framed does not admit beliefs about the future. Since goals and intentions are future-oriented, in the absence of beliefs about the future, the relationship between beliefs on the one hand, and goals and intentions on the other is not very interesting. There are implemented systems (e.g., [5]) that have goals with propositional content, however, as far as we are aware, there are no implemented systems that allow explicit beliefs about the future.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have addressed an important aspect of agent reasoning—intention revision. Intention revision is central, since an agent typically pursues multiple tasks, adopting intentions to achieve them. These intentions may conflict with each other, due to resource limitations, for example. In the event of such conflicts the agent ought to resolve them in a *rational* manner.

We have defined a framework for the revision of sets of intentions that mutually conflict. We have presented two approaches. The first is to revise the set of intentions by dropping one or more intentions to attain a non-conflicting set, and the other is to modify the current intentions so that they may be achieved by alternative means to obtain a non-conflicting set of intentions.

Our theory of intention revision is embedded in a powerful formal framework for the representation of goals, programs, and the environments under which they are executed. This framework allows us to specify rich criteria for appropriate revised sets of intentions to satisfy: *robustness* (guaranteed success in maximal number of environments), minimal reduction on dropping an intention (i.e., preserving maximal number of top-level intentions), and minimal change to the means of achieving an intention.

Future work includes extending our model to include further features of Grant *et al.*'s framework, i.e., adding new intentions, representing costs of action and value of achieving goals. In such an account, we would expect to be able to demonstrate satisfaction of their postulates within our much richer framework of goals and plans, providing a clear specification of intention revision behaviour for BDI agent programming models. Our ultimate aim is a model of the complete intention reconsideration problem, including intention selection, opportunistic merging, and revision; and its application to the design of such processes in practical agent programming frameworks. In this paper, we presented a semantics for intention revision and future work also involves developing an implementation that is faithful to these semantics but also computationally feasible.

7. REFERENCES

[1] R. H. Bordini, L. Braubach, M. Dastani, A. Fallah-Seghrouchni, J. J. Gómez Sanz, J. Leite, G. O'Hare, A. Pokahr, and A. Ricci. A survey of programming languages and platforms for multi-agent systems. *Informatica (Slovenia)*, 30(1):33–44, 2006.

[2] M. E. Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press, 1987.

[3] M. E. Bratman, D. J. Israel, and M. E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(3):349–355, 1988.

[4] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[5] F. S. de Boer, K. V. Hindriks, W. van der Hoek, and J.-J. Meyer. A verification framework for agent programming with declarative goals. *Journal of Applied Logic*, 5(2):277–302, 2007.

[6] D. Dennett. *The Intentional Stance*. MIT Press, 1987.

[7] P. Gärdenfors. *Knowledge in Flux*. The MIT Press, 1988.

[8] J. Grant, S. Kraus, D. Perlis, and M. Wooldridge. Postulates for revising BDI structures. *Synthese*, 175:127–150, 2010.

[9] J. Grant, S. Kraus, and M. Wooldridge. Intentions in equilibrium. In M. Fox and D. Poole, editors, *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 786–791. AAAI Press, 2010.

[10] M. Hennessy. *The Semantics of Programming Languages*. John Wiley & Sons, Chichester, England, 1990.

[11] T. Icard, E. Pacuit, and Y. Shoham. Joint revision of beliefs and intentions. In F. Lin, U. Sattler, and M. Truszczynski, editors, *Proceedings of Principles of Knowledge Representation and Reasoning (KR)*, pages 572–574. AAAI Press, 2010.

[12] S. Owre, S. Rajan, J. M. Rushby, N. Shankar, and M. K. Srivas. PVS: Combining specification, proof checking, and model checking. In R. Alur and T. A. Henzinger, editors, *Proceedings of the International Conference on Computer Aided Verification (CAV)*, volume 1102 of *Lecture Notes in Computer Science (LNCS)*, pages 411–414. Springer-Verlag, 1996.

[13] A. S. Rao. Agentspeak(L): BDI agents speak out in a logical computable language. In W. V. de Velde and J. W. Perram, editors, *Proceedings of the European Workshop on Modelling Autonomous Agents in a Multi Agent World (MAAMAW)*, volume 1038 of *Lecture Notes in Computer Science (LNCS)*, pages 42–55. Springer, 1996.

[14] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In J. F. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of Principles of Knowledge Representation and Reasoning (KR)*, pages 473–484. Morgan Kaufmann, 1991.

[15] A. S. Rao and M. P. Georgeff. An abstract architecture for rational agents. In B. Nebel, C. Rich, and W. R. Swartout, editors, *Proceedings of Principles of Knowledge Representation and Reasoning (KR)*, pages 438–449. Morgan Kaufmann, 1992.

[16] A. S. Rao and M. P. Georgeff. BDI agents: From theory to practice. In V. Lesser and L. Gasser, editors, *Procedings of the International Conference on Multiagent Systems (ICMAS)*, pages 312–319. AAAI Press / MIT Press, 1995.

[17] S. Sardina and L. Padgham. A BDI agent programming language with failure recovery, declarative goals, and planning. *Autonomous Agents and Multi-Agent Systems*, 23(1):18–70, 2011.

[18] W. van der Hoek, W. Jamroga, and M. Wooldridge. Towards a theory of intention revision. *Synthese*, 155(2):265–290, 2007.

[19] W. Wobcke. Plans and the revision of intentions. In C. Zhang and D. Lukose, editors, *Distributed Artificial Intelligence: Architecture and Modelling*, volume 1087 of *Lecture Notes in Computer Science (LNCS)*, pages 100–114. Springer, 1995.

Session 4F
Logics for Agency

Action models for knowledge and awareness

Hans van Ditmarsch
Logic
University of Seville, Spain
hvd@us.es

Tim French
Computer Science and Software Engineering
University of Western Australia
tim@csse.uwa.edu.au

Fernando R. Velázquez-Quesada
Logic
University of Seville, Spain
FRVelazquezQuesada@us.es

ABSTRACT

We consider semantic structures and logics that differentiate between being uncertain about a proposition, being unaware of a proposition, becoming aware of a proposition and getting to know the truth value of a proposition. We give a unified setting to model all this variety of static and dynamic aspects of awareness and knowledge, without any constraints on the modal properties of knowledge (or belief — such as introspection) or on the interaction between awareness and knowledge (such as awareness introspection). Our primitive epistemic operator is called *speculative knowledge*. This is different from the better known *implicit knowledge*, now definable, which plays a more restricted role. Some dynamic semantic primitives that are elegantly definable in our setting are the actions of 'becoming aware of a propositional variable', 'implicit knowledge', 'addressing a novel issue in an announcement', and also more complex ways in which an agent can become aware of a novel issue by way of increasing the complexity of the epistemic model.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Modal Logic*

General Terms

Theory

Keywords

Modal logic, Epistemic logic, Awareness, Dynamics

1. INTRODUCTION

We consider a framework that differentiate between (i) agents being uncertain about the value of a proposition, (ii) agents being unaware of a proposition, (iii) agents becoming aware of propositions, and (iv) agents being informed of the truth of propositions of which they were already aware.

Example 1. *Alfred likes football. He supports the English national football team and he is aware that yesterday there was a match between England and The Netherlands, but he does not know which team won. He does not like other sports, so he is unaware that the English national rugby team played yesterday too. When looking online for the football match's result, Alfred sees a web page with header* "The English team faced a complicated rugby match yesterday"*, hence becoming aware of that match (without getting to know who won). He keeps looking for the score of the football match and finally finds it: England 2 - The Netherlands 1.*

In this paper we give a unified setting to model all this variety of static and dynamic aspects of awareness and knowledge, without any constraints on the modal properties of knowledge (or belief - such as introspection) or on the interaction between awareness and knowledge (such as awareness introspection). Our work is rooted in: the tradition of epistemic logic [11] and in particular multi-agent epistemic logic [13, 4]; in various works on the interaction of awareness and knowledge [3, 14, 15, 9] — including a relation to recent works like [10, 7, 8]; and in modal logical research in propositional quantification, starting in the 1970s with [5] and followed up by work on bisimulation quantifiers [24, 12, 6].

Works treating awareness either follow a *semantically* flavoured approach, where awareness concerns propositional variables in the valuation [15, 9], or a more *syntactically* flavoured approach, where awareness concerns all formulas of the language in a given set, in order to model 'limited rationality' of agents [3, 19]. Our proposal falls straight into the semantic corner: within the limits of their awareness, agents are fully rational. Our proposal extends the work of [20, 21] — these works treat the static interaction of knowledge and awareness but not its dynamics, and in particular not the wide variety of dynamics in action models.

2. STRUCTURES

Our semantic model augments standard epistemic (Kripke) models with a parameter to define the notion of awareness.

Definition 1 (Epistemic awareness model). *Given a countable set of atomic propositions P and a finite set of agents N, where these sets are disjoint, an* epistemic awareness model *is a tuple $M = (S, R, \mathcal{A}, V)$ where*

- S *is the* domain*: a non-empty set of (propositional) states also called* worlds *and also denoted by $\mathcal{D}(M)$;*
- $R : N \to \mathcal{P}(S \times S)$ *is an* accessibility function *assigning to each agent $i \in N$ a binary accessibility relation;*
- $\mathcal{A} : N \to S \to \mathcal{P}(P)$ *is an* awareness function *returning the set of atomic propositions agent $i \in N$ is aware of at state $s \in S$ (agent i's awareness state at s);*

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

- $V : P \to \mathcal{P}(S)$ *is a* valuation function *indicating, for each atomic proposition $p \in P$, the set of states $V(p)$ in which the proposition is true.*

We will write R_i for $R(i)$ and $\mathcal{A}_i$ for $\mathcal{A}(i)$. A pair (M, s) with M an epistemic awareness model and s a state in $\mathcal{D}(M)$, the evaluation state, is an epistemic awareness state.

In epistemic awareness models, awareness is specified by the awareness function. Our notion of awareness is given in terms of a set of atomic propositions, different from the models of general awareness of [3] in which awareness is given in terms of an arbitrary set of formulas.

Just like with epistemic models, we can impose requirements on epistemic awareness models. The standard ones are properties of the accessibility relations, like reflexivity, seriality or transitivity. We do not discuss closure properties of the awareness set $\mathcal{A}_i(s)$ as done in [3] because $\mathcal{A}_i(s)$ is a set of atoms rather than an arbitrary set of formulas. There are also properties that relate the accessibility relations with the awareness function. One interesting example is the property of *awareness introspection* [9], which holds when awareness sets are preserved by the accessibility relation: $p \in \mathcal{A}_i(s)$ implies $p \in \mathcal{A}_i(t)$ for every state t such that $(s, t) \in R_i$. As interesting as such models can be, we make no commitment to any particular property, focussing on the most general class of epistemic awareness models.

A notion of *bisimulation* [16] between epistemic awareness models can be obtained by extending the standard definition with a clause that asks for the awareness function to assign, for every agent, the same set of atomic propositions in bisimilar states. The bisimulation requirements can also be restricted to a subset of atomic propositions; this makes sense in our setting because agents may not be aware of every atom. But we go one step further: agents may be aware of different atomic propositions in different states, so in order to indicate when two epistemic awareness models are indistinguishable from the perspective of an agent (or a set of them), we ask for an additional restriction. The result is called *awareness bisimulation.*

Definition 2 (Awareness bisimulation). *Let $M = (S, R, \mathcal{A}, V)$ and $M' = (S', R', \mathcal{A}', V')$ be two epistemic awareness models. For any $Q \subseteq P$, a relation $\mathfrak{R}[Q] \subseteq (S \times S')$ is called a Q-*awareness bisimulation *between M and M' if, for every $(s, s') \in \mathfrak{R}[Q]$:*

- **atoms:** *for all $p \in Q$, $s \in V(p)$ iff $s' \in V'(p)$;*
- **aware:** *for all $i \in N$, $Q \cap \mathcal{A}_i(s) = Q \cap \mathcal{A}'_i(s')$;*
- **forth:** *for all $i \in N$, if $t \in S$ and $R_i(s, t)$ then there is a $t' \in S'$ such that $R'_i(s', t')$ and $(t, t') \in \mathfrak{R}[Q \cap \mathcal{A}_i(s)]$;*
- **back:** *for all $i \in N$, if $t' \in S'$ and $R'_i(s', t')$ then there is a $t \in S$ such that $R_i(s, t)$ and $(t, t') \in \mathfrak{R}[Q \cap \mathcal{A}'_i(s')]$.*

*We say that (M, s) and (M', s') are Q-*awareness-bisimilar *(notation: $(M, s) \leftrightarroweq^Q (M', s')$) if there is a Q-awareness bisimulation between M and M' that contains (s, s').*

The **aware** clause is the additional 'atomic' requirement, given the nature of our models. The further requirement that distinguishes awareness bisimulation from a restricted bisimulation appears in the **forth** and **back** clauses: instead of being $\mathfrak{R}[Q]$-bisimilar, states t and t' need to be just $\mathfrak{R}[Q \cap \mathcal{A}_i(s)]$-bisimilar and $\mathfrak{R}[Q \cap \mathcal{A}'_i(s')]$-bisimilar, respectively — note that by the **aware** clause, $Q \cap \mathcal{A}_i(s)$ and $Q \cap \mathcal{A}'_i(s')$ are the same. The motivation is very simple: two states are Q-awareness-bisimilar for an agent i if they appear Q-identical to her. Since she does not need to be aware of every atom, the states just have to be identical up to those atoms of Q *the agent is aware of.* Then, for the **atoms** clause, we just need to check that both states coincide in the truth values of atoms in Q. Moreover, in the **forth** clause, the bisimulation for state t is further restricted to the propositions visible for agent i in s, the i-predecessor of t; similarly for **back**. This ensures us that only atoms the agent is aware of at the current state will matter when looking for a difference in accessible worlds. This chaining requirement was present in epistemic awareness structures since its inception in [3].

Awareness bisimulation gives us a form of *observational equivalence* among epistemic awareness models. If an agent i is in state s, then her perspective is that of $\mathcal{A}_i(s)$-awareness-bisimilarity: she cannot distinguish the current model from those that are in its $\mathfrak{R}[\mathcal{A}_i(s)]$ equivalence class. This can be generalized for a set of agents N; their perspective is that of $\bigcup_{i \in N} \mathcal{A}_i(s)$-awareness-bisimilarity, so two epistemic awareness states (M, s) and (M', s') are *observationally equivalent* for the agents in N iff no one can distinguish them, that is, iff they are $\mathcal{A}_i(s)$-awareness bisimilar for all $i \in N$:

$$(M, s) \leftrightarroweq^{\bigcup_{i \in N} \mathcal{A}_i(s)} (M', s') \ .$$

If every agent is aware of every atom at every state, we get standard (restricted) bisimulation: for agents with full awareness we go back to the standard multi-agent epistemic situation, where awareness plays no role.

Example 2. *The diagram below shows three epistemic awareness states, (M, s), (M', s') and (M'', s''). In it, each state shows its name, the truth value it assigns to atoms (the overline indicates falsity) and the awareness set for agent i in the format $i^{\mathcal{A}_i(s)}$; the evaluation states are underlined.*

The states (M, s), (M', s') and (M'', s'') are $\{p\}$-awareness bisimilar (e.g., $\{(s, s''), (t, t''_1), (t, t''_2)\}$ is a $\{p\}$-awareness bisimulation between the first and the third). This is because not only the s-states (s, s' and s'') coincide in the truth value of and in agent i's awareness of every atom in $\{p\}$ (clauses **atoms** *and* **aware** *of the definition — note how the truth value of q is irrelevant), but also because every t-state in each model is $\{p\}$-awareness bisimilar to every t-state in the others (clauses* **back** *and* **forth**, *given that $\{p\} \cap \mathcal{A}_i(s) = \{p\}$). (Indeed, the awareness of i is $\varnothing$ in all four t-states; if it had been $\{p\}$ in all four it would have worked as well.)*

$$\begin{tikzcd}
t(pq, i^{\varnothing}) & t'(p\overline{q}, i^{\varnothing}) & t''_1(pq, i^{\varnothing}) & & t''_2(p\overline{q}, i^{\varnothing}) \\
\underline{s(pq, i^{\{p\}})} \arrow[u, "i"] & \underline{s'(pq, i^{\{p\}})} \arrow[u, "i"] & & \underline{s''(p\overline{q}, i^{\{p\}})} \arrow[ul, "i"'] \arrow[ur, "i"] & \\
\boldsymbol{M} & \boldsymbol{M'} & & \boldsymbol{M''} &
\end{tikzcd}$$

3. ACTION MODELS

Epistemic awareness models allow us to represent the information of agents who may be uncertain of the truth value of atomic propositions and may be even unaware of some of them. But, of course, the information of such agents can change via different informational acts. The general structure that we introduce now, epistemic awareness action models, allow us to represent, as far as we know, *any* conceivable form of awareness change or of knowledge change.

Definition 3 (Epistemic awareness action model). *Let P and N be sets of atomic propositions and agents, respectively, with properties as before. An* epistemic awareness **action** model *is a tuple* $\mathsf{M} = (\mathsf{S}, \mathsf{R}, \mathcal{A}, \mathsf{pre}, \mathsf{post})$ *where*

- S *is a non-empty domain: a set of* actions *also denoted by* $\mathcal{D}(\mathsf{M})$*;*
- $\mathsf{R} : N \to \mathcal{P}(\mathsf{S}\times\mathsf{S})$ *is an* accessibility function*, assigning to each agent* $i \in N$ *an accessibility relation* $\mathsf{R}(i)$*;*
- $\mathcal{A} : \{+,-\} \to N \to \mathsf{S} \to \mathcal{P}(P)$ *is an* awareness **change** function*, indicating the* disjoint sets *of atoms each agent* $i \in N$ *will become aware (+) and unaware of (-) after the execution of* $\mathsf{s} \in \mathsf{S}$*;*
- $\mathsf{pre} : \mathsf{S} \to \mathcal{L}$ *is a* precondition function *that specifies, for each action* $\mathsf{s} \in \mathsf{S}$*, the requirement for its execution;*
- $\mathsf{post} : \mathsf{S} \to P \to \mathcal{L}$ *is a* postcondition function *specifying, for each action in* $\mathsf{s} \in \mathsf{S}$*, how the truth value of each atomic proposition* $p \in P$ *will change.*

A pair (M, s) *with* M *an epistemic awareness action model and* s *an action in* $\mathcal{D}(\mathsf{M})$ *is an* epistemic awareness action.

The language $\mathcal{L}$ in terms of which we specify the preconditions and postconditions is a fixed parameter of this definition. In Section 4 we give an integrated approach for the syntax and semantics of a logical language with epistemic awareness action models, wherein $\mathcal{L}$ is not a fixed parameter. As before, we write R_i for $\mathsf{R}(i)$; also, we write $\mathcal{A}_i^+$ for $\mathcal{A}(+)(i)$ and $\mathcal{A}_i^-$ for $\mathcal{A}(-)(i)$.

We can now indicate how an epistemic awareness action model modifies an epistemic awareness model. The following definition is essentially the *product update* of [1] with an additional clause that deals with awareness.

Definition 4 (Action model execution). *Let* $M = (S, R, \mathcal{A}, V)$ *and* $\mathsf{M} = (\mathsf{S}, \mathsf{R}, \mathcal{A}, \mathsf{pre}, \mathsf{post})$ *be an epistemic awareness model and an epistemic awareness action model, respectively. The epistemic awareness model* $M \otimes \mathsf{M} = (S', R', \mathcal{A}', V')$ *– the result of executing* M *in* M *– is defined as follows:*

$$\begin{aligned}
S' &:= \{(s, \mathsf{s}) \mid (M, s) \models \mathsf{pre}(\mathsf{s})\} \\
R'_i &:= \{((s, \mathsf{s}), (s', \mathsf{s}')) \mid (s, s') \in R_i \text{ and } (\mathsf{s}, \mathsf{s}') \in \mathsf{R}_i\} \\
\mathcal{A}'_i(s, \mathsf{s}) &:= (\mathcal{A}_i(s) \cup \mathcal{A}_i^+(\mathsf{s})) \setminus \mathcal{A}_i^-(\mathsf{s}) \\
V'(p) &:= \{(s, \mathsf{s}) \mid (M, s) \models \mathsf{post}(\mathsf{s}, p)\}
\end{aligned}$$

The new set of states is given by the restricted Cartesian product of S and S: a pair (s, s) will be a state in the new model iff s satisfies s's precondition in M. Since the precondition is given as a formula of a language $\mathcal{L}$, we assume a satisfiability relation $\models$ that indicates whether a formula of $\mathcal{L}$ evaluates to true or false in an epistemic awareness state. For the accessibility relation of the new model, we simply combine the accessibility relation of the 'static' and the 'action' model: a state (s', s') is R'_i-accessible from state (s, s) iff s' is R_i-accessible from s, and s' is R_i-accessible from s. For the awareness function of each agent i in each state (s, s), we add the atoms in $\mathcal{A}_i^+(\mathsf{s})$ and remove the atoms in $\mathcal{A}_i^-(\mathsf{s})$ (in whatever order—we require these sets to be disjoint). Finally, for the valuation, an atomic proposition p is true at state (s, s) iff s satisfies $\mathsf{post}(\mathsf{s}, p)$ in M.

The epistemic awareness *state* that results from executing (M, s) in (M, s) is given by$(M \otimes \mathsf{M}, (s, \mathsf{s}))$ whenever $(M, s) \models \mathsf{pre}(\mathsf{s})$.

Example 3. *Below is the diagram of an epistemic awareness action model. Each one of the actions indicates also its precondition and its awareness change function (the latter with the format* $i^{+\mathcal{A}_i^+, -\mathcal{A}_i^-}$ *for every agent* i*). Here the postcondition function is trivial:* $\mathsf{post}(\mathsf{s})(p) = \mathsf{post}(\mathsf{t})(p) = p$*.*

$$\begin{tikzcd}
\underline{\mathsf{s}\ (p, i^{+\{p\}, -\varnothing})} \arrow[loop left, "i"] \arrow[r, leftrightarrow, "i"] & \mathsf{t}\ (\neg p, i^{+\varnothing, -\varnothing}) \arrow[loop right, "i"]
\end{tikzcd}$$

The only difference between actions s *and* t *is the precondition and the fact that* s *adds* p *to agent* i*'s awareness. This epistemic awareness action model can be seen as a form of 'conditionally becoming aware', where the agent becomes aware of* p *in the states in which* p *holds, and keeps her old awareness in states in which* p *fails (see Definition 13). We will see more examples of epistemic awareness action models and their execution in Section 8.*

4. LANGUAGE

Section 2 presented a semantic structure, epistemic awareness model, for representing the information of agents that do not need to be aware of all the relevant atoms. Then Section 3 introduced another structure, epistemic awareness action model, that allows us to represent diverse actions that can change the agents' information. However, a parameter in these action models was a logical language.

We can also do this 'all at once': an inductively defined language, with an appropriately compositional semantics, wherein a countable set of 'action model shapes' features as a parameter in the language. This language allows us to describe epistemic awareness models and how they change after an epistemic awareness action model is applied.

Definition 5 (Language). *Given sets of atomic propositions* P *and agents* N *as before, the language* $\mathcal{L}$ *of the* logic of knowledge and awareness change *is given by*

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i^S \varphi \mid A_i \varphi \mid [\mathsf{M}, \mathsf{s}]\varphi$$

where $i \in N$*,* $p \in P$ *and* (M, s) *is a epistemic awareness action satisfying that:*

(a) its domain is finite;

(b) the postcondition function changes the valuation of only a finite number of atomic propositions.

(c) the awareness function returns two finite sets of atomic propositions;

The language $\mathcal{L}$ extends multi-agent epistemic logic with two operators. The first, $A_i\varphi$, expresses that agent i *is* aware of φ; the second, $[\mathsf{M}, \mathsf{s}]\varphi$, stands for *"after (every) execution of the epistemic awareness action* (M, s)*,* φ *is the case"*. Implication $\to$, disjunction $\vee$, and equivalence $\leftrightarrow$ are defined by abbreviation as usual, and $L_i^S\varphi$ is defined as $\neg K_i^S \neg\varphi$. Note that $\top$ is explicitly a primitive in the language; we do not define it with an abbreviation of the form $p \vee \neg p$ for some atom p because we want every agent to be aware of $\top$ even if they are unaware of every atomic proposition.

The epistemic operator K_i^S is non-standard. It stands for agent i's *speculative knowledge* (a notion called implicit knowledge in [20]), and its semantic interpretation will be introduced in the next section. Explicit knowledge is defined by $K_i^E\varphi$ iff $K_i^S\varphi \wedge A_i\varphi$ (cf. [3]).

The case $[\mathsf{M}, \mathsf{s}]\varphi$ of the inductive language definition is indeed a proper induction, because it can be seen as an operation on the set of all preconditions $\mathsf{pre}(\mathsf{t})$ of actions in the action model, and the formula φ. In the definition, all these are supposed to be of type formula and lower in the inductive hierarchy. The restrictions on epistemic awareness actions in the language are imposed so that the inductively defined language $\mathcal{L}$ is well-defined—the class of epistemic awareness actions needs to be enumerable, and (as an independent requirement) the number of arguments in the inductive construct $[\mathsf{M}, \mathsf{s}]\varphi$ needs to be finite. For this we need all three finiteness requirements. For (a) this was known since [22]; for (b) this was known since dynamic epistemic logics modelling factual change, namely [18]; we can make our logic tick by the novel requirement (c). If we merely wished a semantic treatment of epistemic awareness actions, the finiteness requirements would not be needed.

As mentioned, our notion of awareness is semantic: an agent is aware of a formula if she is aware of the set of atomic propositions in that formula. For this, we need to define the *free variables* of a formula.

Definition 6 (Free variables). *The free propositional variables of $\varphi \in \mathcal{L}$ are defined inductively in the following way: $v(\top) := \varnothing$; $v(p) := \{p\}$; $v(\neg\varphi) := v(\varphi)$; $v(\varphi \wedge \psi) := v(\varphi) \cup v(\psi)$; $v(K_i^S\varphi) := v(\varphi)$, $v(A_i\varphi) := v(\varphi)$, $v([\mathsf{M}, \mathsf{s}]\varphi) := \bigcup_{\mathsf{t} \in \mathcal{D}(\mathsf{M})} v(\mathsf{pre}(\mathsf{t})) \cup \bigcup_{\mathsf{t} \in \mathcal{D}(\mathsf{M}), p \textit{ changes}} v(\mathsf{post}(\mathsf{t})(p)) \cup v(\varphi)$ where 'p changes' means that $p \in \mathcal{A}_i^+(\mathsf{t})$ or $p \in \mathcal{A}_i^-(\mathsf{t})$ for some agent i.*[1]

Concerning $v([\mathsf{M}, \mathsf{s}]\varphi)$, recall that the modality $[\mathsf{M}, \mathsf{s}]$ represents an inductive case of the language with the preconditions $\mathsf{pre}(\mathsf{t})$, postconditions $\mathsf{post}(\mathsf{t})(p)$ and the formula φ as arguments.

5. SEMANTICS

Having defined the structures and the language to describe them, we now define the semantic interpretation.

Definition 7 (Semantics). *Let $M = (S, R, \mathcal{A}, V)$ and $s \in \mathcal{D}(M)$ be given. The semantics for $\top$, atoms, negation and conjunction is as usual. For the rest,*

$$\begin{array}{lll} (M, s) \models K_i^S\varphi & \textit{iff} & \forall (s,t) \in R_i \textit{ and } \forall (M', t') \underline{\leftrightarrow}^{\mathcal{A}_i(s)} (M, t), \\ & & (M', t') \models \varphi \\ (M, s) \models A_i\varphi & \textit{iff} & v(\varphi) \subseteq \mathcal{A}_i(s) \\ (M, s) \models [\mathsf{M}, \mathsf{s}]\varphi & \textit{iff} & (M, s) \models \mathsf{pre}(\mathsf{s}) \Rightarrow (M \otimes \mathsf{M}, (s, \mathsf{s})) \models \varphi \end{array}$$

The set of validities of the language $\mathcal{L}$ is called the logic L.

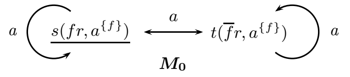
An agent speculatively knows φ when φ remains true in all accessible states for *every* possible interpretation of all propositions she is unaware of. We achieve this by composing the agent's accessibility relation with bisimulation restricted to propositions the agent is aware. This speculative knowledge is not implicit knowledge K^I in the Fagin et al. [3] sense where this corresponds to mere modal accessibility

$$(M, s) \models K_i^I\varphi \quad \text{iff} \quad \forall (s,t) \in R_i, (M, t) \models \varphi$$

[1]The clause for $[\mathsf{M}, \mathsf{s}]\varphi$ makes the semantics of $A_i\varphi$ (below) work but is too restrictive given our intuitions of awareness. For example, given that $[p := q]p \leftrightarrow q$, we want that $v([p := q]p) = v(q) = q$, but our definition gives $\{p, q\}$. This will be further investigated.

Example 4. *Consider the epistemic awareness states of Example 2. In all three cases agent i knows p explicitly but she does not know q explicitly at s because the accessible t is p-awareness bisimilar to (e.g.) t' where q is false. If she becomes aware of q in state s, then she will know q explicitly: any state $\{p, q\}$-awareness bisimilar to t* must *satisfy q.*

Example 5. *Here is an epistemic awareness state for Alfred's situation before his online search (Example 1); we will use formulas of $\mathcal{L}$ to show that it represents the situation faithfully. We will use f (r) to indicate that England won the football (rugby) match.*



$$\begin{tikzcd} \underline{s(fr, a^{\{f\}})} \arrow[loop left, "a"] \arrow[r, leftrightarrow, "a"] & t(\overline{f}r, a^{\{f\}}) \arrow[loop right, "a"] \end{tikzcd}$$

$\boldsymbol{M_0}$

At (M_0, s), Alfred (a) is aware of the football match (f) but unaware of the rugby match (r) because A_af and $\neg A_ar$ hold (f is in $\mathcal{A}_a(s)$ but r is not). Moreover, Alfred does not have any speculative (and hence does not have any explicit) knowledge about who won any of the matches since neither of the following formulas hold at (M_0, s): K_a^Sf, $K_a^S\neg f$, K_a^Sr, $K_a^S\neg r$. This is because for $f, \neg f, r$ and $\neg r$ we can find a state u reachable from s (e.g., t for the first formula; s for the second, third and fourth) and an epistemic awareness state (M', u') that is $\{f\}$-awareness-bisimilar to (M_0, u) and in which the given formula fails. For the first formula, any state $\{f\}$-awareness-bisimilar to (M_0, t) should satisfy $\neg f$; analogously for the second. For the third and fourth, a state $\{f\}$-awareness-bisimilar to (M_0, s) can assign any truth value to r.

Our main result is that in epistemic awareness bisimilar states the agents have the same explicit knowledge; this justifies a more complex form of bisimulation and the notion of speculative knowledge that is more complex than 'standard' (implicit) knowledge. Although there is fairly direct proof of this (consisting of *that* case of the following inductive proof), we present it in the context of a more general, staged, result. Below, we will use the following valid observation. Let $v(\varphi) = Q' \subseteq Q$ and take an epistemic awareness state (M, s): φ is true in all states (M', s') that are Q-bisimilar to (M, s) iff φ is true in all states (M'', s'') that are Q'-bisimilar to (M, s). From left to right the statement holds because we look at less atoms; from right to left it holds because the extra atoms do not appear in φ. In other words, variation in variables *not* occurring in φ does not affect its value. For $Q \setminus Q' = \{p\}$ this corresponds to the validity $\varphi \leftrightarrow \forall p\varphi$ whenever $p \notin v(\varphi)$, in bisimulation quantified logic.

Theorem 8. *If $(M, s) \underline{\leftrightarrow}^Q (M', s')$ and $v(\varphi) \subseteq Q$, then $(M, s) \models \varphi$ iff $(M', s') \models \varphi$.*

Proof. To be more precise, the statement we prove is "Let $\varphi \in \mathcal{L}$. Then for every epistemic awareness states (M, s) and (M', s'), if $(M, s) \underline{\leftrightarrow}^Q (M', s')$ and $v(\varphi) \subseteq Q$, then $(M, s) \models \varphi$ iff $(M', s') \models \varphi$." The proof is by induction on φ, and the cases of interest are $K_i^S\varphi$, $A_i\varphi$, and $[\mathsf{M}, \mathsf{s}]\varphi$.

- **Base case p:** From the **atoms** clause of bisimulation, the fact that $p \in Q$, and $(M, s) \underline{\leftrightarrow}^Q (M', s')$, it follows that $(M, s) \models p$ iff $(M', s') \models p$.
- **Inductive case $\neg\varphi$:** Showing that $(M, s) \models \neg\varphi$ iff $(M', s') \models \neg\varphi$ is equivalent to showing that $(M, s) \not\models \varphi$ iff $(M', s') \not\models \varphi$; swapping the order delivers $(M, s) \models \varphi$ iff $(M', s') \models \varphi$ which follows by induction.

- **Inductive case $\varphi \wedge \psi$:** Let $(M,s) \underline{\leftrightarrow}^Q (M',s')$ and $v(\varphi \wedge \psi) \subseteq Q$ (so $v(\varphi), v(\psi) \subseteq Q$). Now $(M,s) \models \varphi \wedge \psi$ iff $(M,s) \models \varphi$ and $(M,s) \models \psi$, iff (using the induction hypothesis on φ and on ψ) $(M',s') \models \varphi$ and $(M',s') \models \psi$, i.e., $(M',s') \models \varphi \wedge \psi$.

- **Inductive case $K_i^S \varphi$:** Let $(M,s) \underline{\leftrightarrow}^Q (M',s')$ and $v(K_i^S \varphi) \subseteq Q$ (so $v(\varphi) \subseteq Q$ as well). Assume $(M,s) \models K_i^S \varphi$. Take some $t' \in R_i(s')$ and (N',u') such that $(M',t') \underline{\leftrightarrow}^{\mathcal{A}'_i(s')} (N',u')$; we will show that $(N',u') \models \varphi$. Because $v(\varphi) \subseteq Q$ and the observation above it is sufficient to prove this for an arbitrary (N',u') with $(M',t') \underline{\leftrightarrow}^{Q \cap \mathcal{A}'_i(s')} (N',u')$.

  From **back** it follows that there is a $t \in R_i(s)$ such that $(M,t) \underline{\leftrightarrow}^{Q \cap \mathcal{A}'_i(s')} (M',t')$. From $(M,t) \underline{\leftrightarrow}^{Q \cap \mathcal{A}'_i(s')} (M',t')$ and $(M',t') \underline{\leftrightarrow}^{Q \cap \mathcal{A}'_i(s')} (N',u')$ follows $(M,t) \underline{\leftrightarrow}^{Q \cap \mathcal{A}'_i(s')} (N',u')$ (bisimilarity is an equivalence relation).

  From that, the semantics of $K^S \varphi$ and again the observation that we may restrict $\mathcal{A}_i(s')$ bisimilarity to $Q \cap \mathcal{A}'_i(s')$ bisimilarity, it follows immediately that $(N',u') \models \varphi$.

  The other direction is similar. Note that, somewhat surprisingly, we have not used induction in this inductive case of the proof.

- **Inductive case $A_i \varphi$:** Let $(M,s) \underline{\leftrightarrow}^Q (M',s')$ and also $v(A_i \varphi) \subseteq Q$ (so $v(\varphi) \subseteq Q$). Now, $(M,s) \models A_i \varphi$ implies $v(\varphi) \subseteq \mathcal{A}_i(s)$, and since $v(\varphi) \subseteq Q$, we have $v(\varphi) \subseteq \mathcal{A}_i(s) \cap Q$. By the **aware** clause of bisimulation, $\mathcal{A}_i(s) \cap Q = \mathcal{A}'_i(s') \cap Q$, so $v(\varphi) \subseteq \mathcal{A}'_i(s') \cap Q$ and hence $v(\varphi) \subseteq \mathcal{A}'_i(s')$, which implies $(M',s') \models A_i \varphi$. The other direction is similar.

- **Inductive case $[\mathsf{M},\mathsf{s}]\varphi$:** Suppose $(M,s) \models [\mathsf{M},\mathsf{s}]\varphi$. Then $(M,s) \models \mathsf{pre}(\mathsf{s})$ implies $((M \otimes \mathsf{M}), (s,\mathsf{s})) \models \varphi$. By induction we have that $(M,s) \models \mathsf{pre}(\mathsf{s})$ iff $(M',s') \models \mathsf{pre}(\mathsf{s})$. The modal product construction in $(M \otimes \mathsf{M})$ is well-known to be bisimulation preserving (see e.g. the original publication [1]); an easily observable fact when one realizes that pairs in the new accessibility relation require the first argument to be in the accessibility relation in the original model (given $(t,t') \in \mathfrak{R}[Q]$, the induced bisimulation $\mathfrak{R}'[Q]$ on the product is defined as $((t,\mathsf{t}),(t',\mathsf{t})) \in \mathfrak{R}'[Q]$). Of course, for our present logic we also have to satisfy the requirement **aware**. In the model $(M \otimes \mathsf{M})$ the level of awareness $\mathcal{A}_i(t,\mathsf{t})$ is a function of the prior level of awareness $\mathcal{A}_i(t)$ in t and the deleted or added propositional variables $\mathcal{A}_i^+(\mathsf{t})$ and $\mathcal{A}_i^-(\mathsf{t})$. As the prior awareness $\mathcal{A}_i(t)$ is the same in any Q bisimilar state t', and the added or deleted atoms are also the same, the posterior awareness must therefore also be the same for any pairs (t,t) and (t',t) in the Q-bisimulation. Therefore, $((M \otimes \mathsf{M}),(s,\mathsf{s})) \underline{\leftrightarrow}^Q ((M' \otimes \mathsf{M}),(s',\mathsf{s}))$. Now using induction again, we conclude $((M' \otimes \mathsf{M}),(s',\mathsf{s})) \models \varphi$, and from that and $(M',s') \models \mathsf{pre}(\mathsf{s})$ we conclude $(M',s') \models [\mathsf{M},\mathsf{s}]\varphi$.

□

Corollary 9. *If $(M,s) \underline{\leftrightarrow}^P (M',s')$, then $(M,s) \models \varphi$ iff $(M',s') \models \varphi$.*

Proof. *Apply Theorem 8 with $Q = P$.* □

Corollary 10. *Epistemic awareness bisimilar states coincide in K^E. For $i \in N$ and $\varphi \in \mathcal{L}$, if $(M,s) \models K_i^E \varphi$ and $(M,s) \underline{\leftrightarrow}^{\mathcal{A}^i(s)} (M',s')$ then $(M',s') \models K_i^E \varphi$.*

Proof. *Apply Theorem 8 with $Q = \mathcal{A}_i(s)$, also using that $v(K_i^E \varphi) = v(\varphi) \subseteq \mathcal{A}_i(s)$.* □

This is a good moment to point out that if we define explicit knowledge in the 'standard' way, namely as awareness plus modal accessibility: $K_i^{EX} \varphi \leftrightarrow (K^I \varphi \wedge A_i \varphi)$ [3], then Corollary 10 fails. Epistemic awareness states that are bisimilar up to the awareness of a given agent do not need to provide the agent the same explicit knowledge in the awareness plus modal accessibility sense.

Example 6. *Consider the following two models:*

$$\begin{tikzcd}
\boldsymbol{M}\colon s(p, i^{\{p\}}) \arrow[r, "i"] & t(p, i^{\varnothing}) \arrow[r, "i"] & u(p, i^{\varnothing}) \\
\boldsymbol{M'}\colon s'(p, i^{\{p\}}) \arrow[r, "i"] & t'(p, i^{\varnothing}) \arrow[r, "i"] & u'(\overline{p}, i^{\varnothing})
\end{tikzcd}$$

Model M has domain $\{s,t,u\}$, a single agent with accessibility relation $R = \{(s,t),(t,u)\}$, atom p true everywhere, and the agent is aware of p only in s. Model M' is like M except that p is false in u'. The only difference between M and M' is therefore p's truth value on the u's states. Observe how (M,u) and (M',u') are $\varnothing$-bisimilar; then, because of this and because the t's states coincide in p's truth value and in i's awareness of p, the epistemic awareness states (M,t) and (M',t') are $\{p\}$-awareness bisimilar. This and the fact that s and s' coincide p's truth value and in i's awareness of p makes the epistemic awareness states (M,s) and (M',s') $\{p\}$-awareness bisimilar too. Nevertheless, we can find formulas with atoms in $\{p\}$ that are known explicitly (in the awareness plus modal accessibility sense) at (M,s) but not at (M',s'). Formula $K^I p$ is an example, as $K^{EX} K^I p$ holds at the first, but not at the second.

Of course this does not say that [3] is incorrect: their setting is for $KD45$ models and then our counterexample, which does not satisfy transitivity, does not work there.

6. PARTIAL AXIOMATIZATION

The logic L is partially axiomatized by the axioms and rules of Table 1. The complete axiomatization in [21] can be seen as the special case for the action model encoding 'all agents become aware of atom p in the entire model', see Definition 13, below. The complete axiomatization of standard action model logic [1] might be seen as the special case for actions that only change knowledge but not awareness, see Section 7, below.

The table starts with axioms and rules for the 'static' part of the language, that is, the one that does not involve either awareness operators or epistemic awareness action modalities [13]. Then we have axioms for speculative knowledge, describing how this operator carries features of bisimulation: if an agent is unaware of an atom, he does not refute any interpretation of that atom, nor does he refute the interpretation of any other agent's awareness of that atom [21].

The next part of the table characterizes the behaviour of the awareness operator. The first five axioms are all standard for atom-based awareness [3, 21]; for the last, the one concerning $A_i[\mathsf{M},\mathsf{s}]\varphi$, the right-hand side merely expresses that the agent has to be aware of all variables in all preconditions of actions in the action model, and also of the variables in φ.

The last part of the table contains reduction axioms for the epistemic awareness actions, relating the truth value of formulas *after* an action to the truth value of formulas *before* it. The one for $\top$ indicates that after any successful execution of (M, s), $\top$ is the case. That for atomic propositions is inherited from [18], and indicates that after an execution of (M, s) the atom p will be true iff, provided it satisfies s's precondition, the current state satisfies (s, p)'s postcondition. The axioms for $\neg$ and $\wedge$ are standard [1].

The axioms for awareness after an epistemic awareness action are novel; they take into account that the level of awareness may increase or decrease after action execution, which gives us two cases. If φ contains an atom the agent becomes unaware of (that is, an atom in $\mathcal{A}_i^-(\mathsf{s})$), then she will not be aware of φ after the epistemic awareness action. Otherwise, after the action the agent will become aware of φ iff she is already aware of φ or if she will be after becoming aware of the atoms in $\mathcal{A}_i^+(\mathsf{s})$, that is, iff she was aware of $\varphi[\top \backslash \mathcal{A}_i^+(\mathsf{s})]$, where $\psi[\chi \backslash \{p_1, \dots, p_n\}]$ is the simultaneous substitution of χ for all occurrences of every $p_1, \dots, p_n$ in ψ.

We have not found axioms for speculative knowledge after an epistemic awareness action. The form $[\mathsf{M}, \mathsf{s}]K_i^I\varphi \leftrightarrow (\mathsf{pre}(\mathsf{s}) \rightarrow \bigwedge_{(\mathsf{s},\mathsf{t}) \in \mathsf{R}_i} K_i^I[\mathsf{M}, \mathsf{t}]\varphi)$ for modal accessibility [1] does not hold for speculative knowledge $K_i^S\varphi$. If $[\mathsf{M}, \mathsf{t}]$ does not change awareness, the axiom still holds, but the problem with an axiom for K_i^S of that form is that the level of awareness for agent i before and after action execution may be different. Surely, if p is true in all accessible states, we want for i to know p explicitly after becoming aware of p (and if that is the *only* dynamics — action models with factual change might after all change the value of p). But we do not want for i to know that before becoming aware of p (neither explicitly, nor speculatively). Dually, if i knows p explicitly, then after becoming unaware of p (as a conscious abstraction action, so to speak), she should no longer explicitly know that. The axioms for the interaction of awareness dynamics and speculative knowledge in [21] suggest that the interaction axioms for speculative knowledge and epistemic awareness actions will *not* be *reduction* axioms, i.e., they will not be equivalences helping us to rewrite formulas into formulas without epistemic awareness actions. Therefore, we may also have to face expressivity questions.

It is worthwhile to note that, though an axiomatization would add value to our framework, it would be mainly for logicians. It is often said that the value of Hilbert-style complete axiomatizations is limited for the multi-agent system applications. Similar logics are undecidable, so it is unclear if effective procedures can be found.

7. TYPES OF EPISTEMIC AWARENESS ACTIONS

We can distinguish different types of epistemic awareness actions, in which we recognize a number of actions familiar from the literature, but also novel additions.

Knowledge change without awareness change To model pure knowledge (or belief) change, the awareness functions $\mathcal{A}^+$ and $\mathcal{A}^-$ should neither add nor delete propositional variables for any agent.

Definition 11 (No awareness change). *An epistemic awareness action model* M *is of type* 'no awareness change' *if for any agent* i *and action* $\mathsf{s} \in \mathcal{D}(\mathsf{M})$, $\mathcal{A}_i^+(\mathsf{s}) = \mathcal{A}_i^-(\mathsf{s}) = \varnothing$.

All propositional tautologies	From $\varphi \rightarrow \psi$ and φ infer ψ
$K_i^S(\varphi \rightarrow \psi) \rightarrow (K_i^S\varphi \rightarrow K_i^S\psi)$	From φ infer $K_i^S\varphi$
$\big((K_i^S(p \rightarrow \varphi) \vee K_i^S(\neg p \rightarrow \varphi)) \wedge \neg A_i p\big) \rightarrow K_i^S\varphi$	if $p \notin v(\varphi)$
$\big((K_i^S(A_j p \rightarrow \varphi) \vee K_i^S(\neg A_j p \rightarrow \varphi)) \wedge \neg A_i p\big) \rightarrow K_i^S\varphi$	if $p \notin v(\varphi)$
$A_i\top$	$A_i\neg\varphi \leftrightarrow A_i\varphi$
$A_i(\varphi \wedge \psi) \leftrightarrow (A_i\varphi \wedge A_i\psi)$	$A_iK_i\varphi \leftrightarrow A_i\varphi$
$A_iA_j\varphi \leftrightarrow A_i\varphi$	
$A_i[\mathsf{M}, \mathsf{s}]\varphi \leftrightarrow (\bigwedge_{\mathsf{t} \in \mathcal{D}(\mathsf{M})} A_i\mathsf{pre}(\mathsf{t}) \wedge A_i\varphi)$	
$[\mathsf{M}, \mathsf{s}]\top \leftrightarrow \top$	
$[\mathsf{M}, \mathsf{s}]p \leftrightarrow \big(\mathsf{pre}(\mathsf{s}) \rightarrow \mathsf{post}(\mathsf{s}, p)\big)$	
$[\mathsf{M}, \mathsf{s}]\neg\varphi \leftrightarrow \big(\mathsf{pre}(\mathsf{s}) \rightarrow \neg[\mathsf{M}, \mathsf{s}]\varphi\big)$	
$[\mathsf{M}, \mathsf{s}](\varphi \wedge \psi) \leftrightarrow \big([\mathsf{M}, \mathsf{s}]\varphi \wedge [\mathsf{M}, \mathsf{s}]\psi\big)$	
$[\mathsf{M}, \mathsf{s}]A_i\varphi \leftrightarrow \neg\mathsf{pre}(\mathsf{s})$	if $v(A_i\varphi) \cap \mathcal{A}_i^-(\mathsf{s}) \neq \varnothing$
$[\mathsf{M}, \mathsf{s}]A_i\varphi \leftrightarrow (\mathsf{pre}(\mathsf{s}) \rightarrow A_i\varphi[\top \backslash \mathcal{A}_i^+(\mathsf{s})])$	otherwise
From φ infer $[\mathsf{M}, \mathsf{s}]\varphi$	

Table 1: Axiom system

This way, we recapture all 'standard' action models à la [1], modulo the additional information in structures on the static awareness of agents.

Awareness change without knowledge change To model pure awareness change, no agent should learn any non-trivial formula. This is guaranteed if in the action model, in any equivalence class for any agent, the disjunction of the preconditions of these actions are equivalent to the triviality.

Definition 12 (No knowledge change). *An epistemic awareness action model* M *is of type* 'no knowledge change' *if for any agent* $i \in N$, *and for all* $\mathsf{s} \in \mathcal{D}(\mathsf{M})$

$$\bigvee_{\mathsf{t} \in \mathsf{s}R_i} \mathsf{pre}(\mathsf{t}) \leftrightarrow \top$$

As awareness change is central to this contribution, let us investigate even more specific ways of becoming aware without knowledge change.

Definition 13 (Conditionally becoming aware). *The epistemic awareness action* $(\mathsf{A}_i^{+p}(\varphi), \mathsf{t})$, *wherein agent* i *becomes aware of* p *in the states satisfying* φ, *has two actions indistinguishable from one another,* t *and* f. *The precondition function is given by* $\mathsf{pre}(\mathsf{t}) = \varphi$ *and* $\mathsf{pre}(\mathsf{f}) = \neg\varphi$, *with* post *the trivial assignment for both actions:* $\mathsf{post}(\mathsf{t})(q) = \mathsf{post}(\mathsf{f})(q) = q$ *for every* $q \in P$ *(including* p*). For the awareness change function only* p *is added, only in* t, *and no atom is removed:* $\mathcal{A}_i^+(\mathsf{t}) = \{p\}$, $\mathcal{A}_i^+(\mathsf{f}) = \mathcal{A}_i^-(\mathsf{t}) = \mathcal{A}_i^-(\mathsf{f}) = \varnothing$.

Other epistemic awareness actions, like $(\mathsf{A}_i^{+Q}(\varphi), \mathsf{t})$, an action wherein i becomes aware of all variables in $Q \subseteq P$, or $(\mathsf{A}_i^{+p}(\top), \mathsf{t})$, the one in which i becomes aware of p in *all* states of the epistemic awareness model, can be defined in a similar way. For the latter we can also use a one-action epistemic awareness action model, because the other action now has a precondition that is never satisfied $(\neg\top)$.

Another interesting case is the one in which *every* agent becomes aware of p. We can model this by either the composition $\mathsf{A}_{i_1}^{+p}(\varphi); \dots; \mathsf{A}_{i_n}^{+p}(\varphi)$ when we have $N = \{i_1, \dots, i_n\}$, or, more elegantly, as an epistemic awareness action $(\mathsf{A}_N^{+p}(\varphi), \mathsf{t})$ wherein $\mathcal{A}_i^+(\mathsf{t}) = \{p\}$ for *every* agent $i \in N$. Then, $\mathsf{A}_N^{+p}(\top)$ represents an action wherein every agent becomes aware of p in the entire model. (This is the special case mentioned at the start of Section 6.)

On finite epistemic awareness models, where each state s has a *distinguishing formula* δ_s only true in (the bisimulation class of) that state [2, 17], we can define an epistemic awareness action with which agent i becomes aware of p only in the actual state:

$$\mathsf{A}_i^{+p}(\delta_{\mathsf{t}}) \ .$$

With the action model for becoming aware, implicit knowledge K^I in the Fagin et al. modal accessibility sense [3] is now definable. (Otherwise, it is not.)

Definition 14 (Implicit knowledge).

$$K_i^I \varphi \quad \textit{iff} \quad [\mathsf{A}_i^{+v(\varphi)}(\top)] K_i^S \varphi \ .$$

If the agent is aware of all variables in φ, $A_i\varphi$ holds, so that speculative knowledge $K_i^S\varphi$ now entails explicit knowledge $K_i^E\varphi$. In other words, the definition spells out that an agent implicitly knows φ (in the sense that φ is true in all i-accessible states) iff after becoming aware of all the variables in φ, she explicitly knows φ.

We should point out however, that in our framework this definition is of limited use, as we also allow for more involved ways of becoming aware, wherein the value of atoms that the agent is unaware of may change (see the next section for a detailed example). In such cases, p may now be true in all accessible states, so you may implicitly know p but after the epistemic awareness action, you explicitly know that p is false, because its value changed in the execution of the action. The reason to permit such actions is to allow for the epistemic complexity of states to increase, thus reflecting the growing knowledge of agents about the atoms they are aware of. Before they were aware of these atoms, there was no need for such complexities and their values could therefore be considered 'don't care' values.

Addressing a novel issue A typical conversational act is an announcement wherein a novel issue is being addressed. Such announcements makes the addressed agents aware and knowledgeable at the same time. Such announcements have been differently modelled in [23].

Definition 15 (Addressing a novel issue). *The* awareness announcement

$$!_A\varphi$$

is the composition of the standard announcement $!\varphi$ *with the action* $\mathsf{A}_N^{+v(\varphi)}(\top)$ *that makes every agent aware all variables in* φ*, or, alternatively and equivalently, defined directly as the singleton epistemic awareness action consisting of domain* s*, accessible to all agents, with* $\mathsf{pre}(\mathsf{s}) = \varphi$*, trivial postcondition* $\mathsf{post}(\mathsf{s})(p) = p$ *for every* $p \in P$*, and such that* $\mathcal{A}_i^+(\mathsf{s}) = v(\varphi)$ *and* $\mathcal{A}_i^-(\mathsf{s}) = \varnothing$ *for all agents.*

In the definition above, in the variant defined by composition, it is imperative that $\mathsf{A}_N^{+v(\varphi)}(\top)$ is *after* the announcement, not before. After all, a truthful announcement to all could be

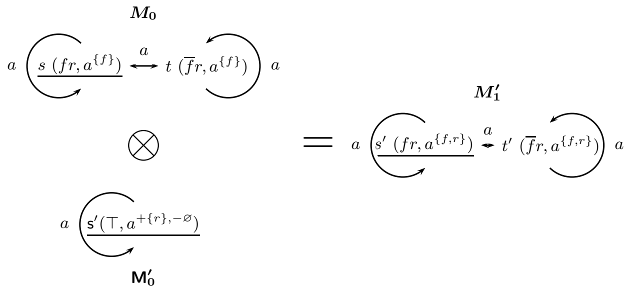
> "You are not aware of the fact that Valencia oranges mature in November!"

This is true at the moment of the announcement, but after that no longer: all have now become aware. We have just discovered the *unsuccessful awareness update*, an announcement to i of the archetypical form $!_A(p \wedge \neg A_i p)$.

8. DETAILED EXAMPLE

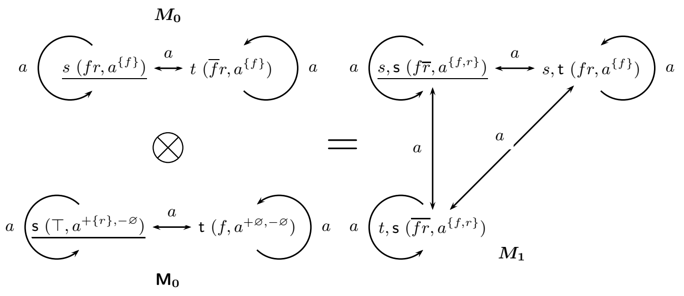
We have shown how Alfred's initial situation (Example 1) can be represented with an epistemic awareness model and described with the language $\mathcal{L}$ (Example 5). Now we will show how his online search and other actions can be represented with action models, and described with modalities expressing the actions' effect.

By reading *"The English team faced a complicated rugby match yesterday"*, Alfred becomes aware of the rugby match. One could think that this act is represented by an epistemic awareness action with a single reflexive action s' and awareness change, precondition and postcondition functions given by $\mathcal{A}_a^+(\mathsf{s}') := \{r\}$ and $\mathcal{A}_a^-(\mathsf{s}') := \varnothing$, $\mathsf{pre}(\mathsf{s}') := \top$ and $\mathsf{post}(\mathsf{s}')(p) := p$ for every atom p, respectively, but this is not the case.


M_0
a
$s\ (fr, a^{\{f\}})$
a
$t\ (\bar{f}r, a^{\{f\}})$
a
M_1'
a
$s'\ (fr, a^{\{f,r\}})$
a
$t'\ (\bar{f}r, a^{\{f,r\}})$
a
a
$\mathsf{s}'(\top, a^{+\{r\},-\varnothing})$
M_0'

Observe how, in the resulting epistemic awareness state, (M_1', s'), Alfred is indeed aware of the rugby match ($A_a r$ holds) but he also knows speculatively that England won it ($K_a^S r$). (Equivalently, $[\mathsf{M}_0', \mathsf{s}'](A_a r \wedge K_a^S r)$ holds at (M_0, s).) This is because every epistemic awareness state that is $\{f, r\}$-awareness bisimilar to either (M_1', s') or else (M_1', t') must satisfy r.

The act of becoming aware of r without getting to know its truth value is represented by the epistemic awareness action $(\mathsf{M}_0, \mathsf{s})$ below, with postconditions given by $\mathsf{post}(\mathsf{s})(f) = \mathsf{post}(\mathsf{t})(f) = f$, $\mathsf{post}(\mathsf{s})(r) = \neg\top$ and $\mathsf{post}(\mathsf{t})(r) = \top$.


M_0
a
$s\ (fr, a^{\{f\}})$
a
$t\ (\bar{f}r, a^{\{f\}})$
a
a
$s, \mathsf{s}\ (f\bar{r}, a^{\{f,r\}})$
a
$s, \mathsf{t}\ (fr, a^{\{f\}})$
a
a
a
a
$\mathsf{s}\ (\top, a^{+\{r\},-\varnothing})$
a
$\mathsf{t}\ (f, a^{+\varnothing,-\varnothing})$
a
a
$t, \mathsf{s}\ (\bar{f}\bar{r}, a^{\{f,r\}})$
M_1
M_0

Now in the resulting epistemic awareness state, (M_1, s, s), Alfred is aware of the rugby match ($A_a r$) without knowing speculatively who won it ($\neg K_a^S r \wedge \neg K_a^S \neg r$). (Equivalently, $[\mathsf{M}_0, \mathsf{s}](A_a r \wedge \neg K_a^S r \wedge \neg K_a^S \neg r)$ holds at (M_0, s).) This is because state (t, s) is reachable from (s, s) and we can find an epistemic awareness state that is $\{f, r\}$-awareness bisimilar to (M_1, t, s) and in which r fails (so $\neg K_a^S r$ holds), and because state (s, t) is reachable from (s, s) and we can find an epistemic awareness state that is $\{f, r\}$-awareness bisimilar to (M_1, s, t) and in which $\neg r$ fails (so $\neg K^S_a \neg r$ holds). Note how $(\mathsf{M}_0, \mathsf{s})$ is an action of type 'no knowledge change' (Definition 12) because the disjunction of the preconditions of all the actions, $\top \vee f$, is equivalent to $\top$.

Since Alfred's main concern is football, he keeps looking until he finds out that England defeated The Netherlands. In this act there is no change in awareness; this is a simple announcement of f in the classical sense.

$$\begin{tikzcd}
\boldsymbol{M_1} \otimes & \underline{\mathsf{s}(f, a^{+\varnothing,-\varnothing})} \arrow[loop above, "a"] & = & \underline{s,\mathsf{s}\ (f\overline{r}, a^{\{f,r\}})} \arrow[loop above, "a"] \arrow[r, leftrightarrow, "a"] & s,\mathsf{t}\ (fr, a^{\{f\}}) \arrow[loop above, "a"] \\
& \mathsf{M_1} & & \boldsymbol{M_2} &
\end{tikzcd}$$

Now Alfred knows explicitly (i.e., is aware of and knows speculatively) that England won the football match. This is expressed equivalently by the following facts: $K^E f$ holds at (M_2, s, s), $[\mathsf{M}_1, \mathsf{s}]K^E f$ holds at (M_1, s, s) and $[\mathsf{M}_0, \mathsf{s}][\mathsf{M}_1, \mathsf{s}]K^E f$ holds at (M_0, s). Note how the epistemic awareness action $(\mathsf{M}_1, \mathsf{s})$ is of type 'no awareness change' (Definition 11).

9. CONCLUSION, FURTHER RESEARCH

We presented a unified setting to model all static and dynamic aspects of awareness and knowledge, without any constraints on the modal properties of knowledge or on the interaction between awareness and knowledge. For this, we needed a primitive epistemic operator called *speculative knowledge*, that is different from *implicit knowledge*. Common awareness dynamics is elegantly definable in our setting, e.g.: 'an agent becoming aware of a propositional variable', 'implicit knowledge', 'addressing a novel issue in an announcement', and also more complex ways in which an agent can become aware of a novel issue by way of increasing the complexity of the epistemic model.

A complete axiomatization is still lacking, as are more detailed investigations of special modal classes, such as $S5$ and $KD45$ and how this influences the axiomatization, and compares to other approaches of awareness change that restrict themselves to such classes. Another future direction is to go from bisimulation to simulation: we expect that within reasonable restrictions (e.g., finite models) execution of an epistemic awareness action models is an *awareness simulation* of the initial epistemic awareness state, and vice versa. This would open the window to more succinct axiom systems and complexity results, and provide corroboration that our language is a suitable and adequate formalization for *any conceivable change of awareness or information.*

Acknowledgements We thank the anonymous AAMAS reviewers for their constructive and helpful criticisms. Hans van Ditmarsch is also affiliated to IMSc (Institute of Mathematical Sciences), Chennai, as associate researcher.

10. REFERENCES

[1] A. Baltag, L. Moss, and S. Solecki. The logic of public announcements, common knowledge, and private suspicions. In I. Gilboa, editor, *TARK '98*, pages 43–56, 1998.

[2] M. Browne, E. Clarke, and O. Grümberg. Characterizing Kripke structures in temporal logic. In H. Ehrig, R. Kowalski, G. Levi, and U. Montanari, editors, *TAPSOFT '87*, pages 256–270. Springer, 1987. LNCS 249.

[3] R. Fagin and J. Halpern. Belief, awareness, and limited reasoning. *Artificial Intelligence*, 34(1):39–76, 1988.

[4] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[5] K. Fine. Propositional quantifiers in modal logic. *Theoria*, 36(3):336–346, 1970.

[6] T. French. *Bisimulation quantifiers for modal logic*. PhD thesis, University of Western Australia, 2006.

[7] D. Grossi and F. R. Velázquez-Quesada. *Twelve Angry Men*: A study on the fine-grain of announcements. In X. He, J. Horty, and E. Pacuit, editors, *LORI '09*, pages 147–160. Springer, 2009. LNCS 5834.

[8] J. Halpern and L. Rego. Reasoning about knowledge of unawareness. *Games and Economic Behavior*, 67(2):503–525, 2009.

[9] A. Heifetz, M. Meier, and B. Schipper. Interactive unawareness. *Journal of Economic Theory*, 130:78–94, 2006.

[10] B. Hill. Awareness dynamics. *Journal of Philosophical Logic*, 39(2):113–137, 2010.

[11] J. Hintikka. *Knowledge and Belief*. Cornell University Press, Ithaca, NY, 1962.

[12] M. Hollenberg. *Logic and bisimulation*. PhD thesis, University of Utrecht, 1998.

[13] J.-J. Meyer and W. van der Hoek. *Epistemic Logic for AI and Computer Science*. CUP, 1995.

[14] S. Modica and A. Rustichini. Awareness and partitional information structures. *Theory and Decision*, 37:107–124, 1994.

[15] S. Modica and A. Rustichini. Unawareness and partitional information structures. *Games and Economic Behavior*, 27:265–298, 1999.

[16] C. Stirling. The joys of bisimulation, 1998. LNCS 1450.

[17] J. van Benthem. Dynamic odds and ends. ILLC Research Report ML-1998-08, 1998.

[18] J. van Benthem, J. van Eijck, and B. Kooi. Logics of communication and change. *Information and Computation*, 204(11):1620–1662, 2006.

[19] J. van Benthem and F. R. Velázquez-Quesada. The dynamics of awareness. *Synthese (Knowledge, Rationality and Action)*, 177(Supplement 1):5–27, 2010.

[20] H. van Ditmarsch and T. French. Awareness and forgetting of facts and agents. In *Proceedings of WI-IAT Workshops 2009*, pages 478–483. IEEE Press, 2009.

[21] H. van Ditmarsch and T. French. Becoming aware of propositional variables. In M. Banerjee and A. Seth, editors, *ICLA '11*, pages 204–218. Springer, 2011. LNCS 6521.

[22] H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*. Springer, 2007.

[23] F. R. Velázquez-Quesada. *Small steps in dynamics of information*. PhD thesis, University of Amsterdam, 2011. ILLC Dissertation Series DS-2011-02.

[24] A. Visser. Bisimulations, model descriptions and propositional quantifiers, 1996. Logic Group Preprint Series 161, Utrecht University.

Epistemic Coalition Logic: Completeness and Complexity

Thomas Ågotnes
Dept of Information Science and Media Studies
University of Bergen
PB. 7802, 5020 Bergen, Norway
thomas.agotnes@infomedia.uib.no

Natasha Alechina
School of Computer Science
University of Nottingham
Nottingham NG8 1BB, UK
nza@cs.nott.ac.uk

ABSTRACT

Coalition logic is currently one of the most popular logics for multi-agent systems. While logics combining coalitional and epistemic operators have received considerable attention, completeness results for epistemic extensions of coalition logic have so far been missing. In this paper we provide several such results and proofs. We prove completeness for epistemic coalition logic with common knowledge, with distributed knowledge, and with both common and distributed knowledge, respectively. Furthermore, we completely characterise the complexity of the satisfiability problem for each of the three logics.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; I.2.4 [**Knowledge representation formalisms and methods**]

General Terms

Theory

Keywords

Epistemic logic, coalition logic, completeness, computational complexity

1. INTRODUCTION

Logics of coalitional ability such as *Coalition Logic* ($\mathcal{CL}$) [17], *Alternating-time Temporal Logic* ($\mathcal{ATL}$) [1], and STiT logics [2], are arguably one of the most popular types of logics in multi-agent systems research in recent years. Many different variants of these logics have been proposed and studied. Most of the obtained meta-logical results have been about computational complexity and expressive power. Completeness results have been harder to obtain, with Goranko's and van Drimmelen's completeness proof for $\mathcal{ATL}$ [8], Pauly's completeness proof for $\mathcal{CL}$ [17] and Broersen and colleagues' completeness proofs for different variants of STiT logic [4, 3, 12] being notable exceptions.

The main construction in coalitional ability logics is of the form $[G]\phi$, where G is a set of agents and ϕ a formula, intuitively meaning that G is *effective* for ϕ, or that G can make ϕ come true no matter what the other agents do. One of the most studied extensions of basic coalitional ability logics is adding *knowledge* operators of the type found in *epistemic logic* [5, 16]: both *individual* knowledge operators K_i where i is an agent, and different types of *group* knowledge operators E_G, C_G and D_G where G is a group of agents, standing for everybody-knows, common knowledge and distributed knowledge, respectively. Combining coalitional ability operators and epistemic operators in general and group knowledge operators in particular lets us express many potentially interesting properties of multi-agent systems, such as [19]:

- $K_i\phi \rightarrow [\{i\}]K_j\phi$: i can communicate her knowledge of ϕ to j
- $C_G\phi \rightarrow [G]\psi$: common knowledge in G of ϕ is sufficient for G to ensure that ψ
- $[G]\psi \rightarrow D_G\phi$: distributed knowledge in G of ϕ is necessary for G to ensure that ψ
- $D_G\phi \rightarrow [G]E_G\phi$: G can cooperate to make distributed knowledge explicit

In this paper we study axiomatisation and complexity of variants of *epistemic coalition logic* ($\mathcal{ECL}$), extensions of coalition logic with individual knowledge and different combinations of common knowledge and distributed knowledge. Coalition logic, the next-time fragment of $\mathcal{ATL}$, is one of the most studied coalitional ability logics, and this paper settles a key problem: completeness of its standard epistemic extensions with group knowledge. We furthermore completely characterise the computational complexity of the satisfiability problem for these extensions.

The combinations of coalitional ability operators and epistemic operators in the logics we study in this paper are *independent*; the original semantics of the operators is not changed. It is well known [14, 13] that there are several interesting variants of "ability" under imperfect knowledge; e.g., being able to achieve something without knowing it, vs. knowing *that* one is able to achieve something but not necessarily knowing *how*, vs. knowing *how* one can achieve something. While the two former examples can be expressed with combinations of operators with standard semantics ($[\{i\}]\phi \wedge \neg K_i[\{i\}]\phi$ and $K_i[\{i\}]\phi$ respectively, in the case of a single agent), in order to be able to express the latter (knowledge of ability "de re"), operators with alternative semantics are needed [14, 18, 11, 13]. We do not consider such operators in the current paper. Even though $\mathcal{ECL}$ with standard semantics cannot express knowledge of ability "de re", it can

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

express many other interesting properties (including the examples above as well as the other "variants" of ability under imperfect knowledge).

While epistemic coalitional ability logics have been studied to a great extent, we are not aware of any published completeness results for such logics with all epistemic operators. [19] gives some axioms of $\mathcal{ATEL}$, $\mathcal{ATL}$ extended with epistemic operators, but does not attempt to prove completeness[1]. Broersen and colleagues [3, 12] prove completeness of variants of STiT logic that include individual knowledge operators, but not group knowledge operators, and [12] concludes that adding group operators is an important challenge.

The rest of the paper is organised as follows. In the next section we first give a brief review of coalition logic, and how it is extended with epistemic operators. We then, in each of the three following sections, consider basic epistemic coalition logic with individual knowledge operators extended with common knowledge, with distributed knowledge, and with both common and distributed knowledge, respectively. For each of these cases we show a completeness result. The reason that we consider each of these three systems separately, rather than only the most expressive logic with both common and distributed knowledge, is first, that we want to carefully chart the results for different combinations of operators (a common practice, also in epistemic logic), and, second, that separate proofs for the common and distributed knowledge cases are useful for further extensions for logics with only these epistemic operators. In Section 6 we consider the computational complexity of the three systems. We conclude in Section 7.

2. BACKGROUND

We will define several extensions of propositional logic, and the usual derived connectives, such as $\phi \to \psi$ for $\neg\phi \vee \psi$, will be used. We will also define a number of axiomatic systems S, and by $\vdash_S \phi$ we mean that the formula ϕ is derivable in system S.

2.1 Coalition Logic

We give a brief overview of Coalition Logic ($\mathcal{CL}$) [17]. Assume a set Θ of atomic propositions, and a finite set N of agents. A *coalition* is a set $G \subseteq N$ of agents. We sometimes abuse notation and write a singleton coalition $\{i\}$ as i.

The language of $\mathcal{CL}$ is defined by the following grammar:

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid [G]\phi$$

where $p \in \Theta$ and $G \subseteq N$.

A *coalition model* is a tuple

$$M = \langle S, E, V \rangle$$

where

- S is a non-empty set of *states*;
- V is a *valuation function*, assigning a set $V(s) \subseteq \Theta$ to each state $s \in S$;
- E assigns a *truly playable effectivity function* (see below) $E(s)$ over N and S to each state $s \in S$.

[1]In an unpublished abstract of a talk given at the LOFT workshop in 2004 [7], the authors propose an axiomatisation of $\mathcal{ATEL}$ with individual knowledge and common knowledge operators. However, a completeness result or proof has not been published (personal communication, Valentin Goranko).

An *effectivity function* [17] over N and a set of states S is a function E that maps any coalition $G \subseteq N$ to a set of sets of states $E(G) \subseteq 2^S$. An effectivity function is *truly playable* [17, 6] iff it satisfies the following conditions (when $X \subseteq S$, $\overline{X}$ denotes the complement $S \setminus X$):

E1 $\forall s \in S \forall G \subseteq N : \emptyset \notin E(G)$ (Liveness)

E2 $\forall s \in S \forall G \subseteq N : S \in E(G)$ (Safety)

E3 $\forall s \in S \forall X \subseteq S : \overline{X} \notin E(\emptyset) \Rightarrow X \in E(N)$ (N-maximality)

E4 $\forall s \in S \forall G \subseteq N \forall X \subseteq Y \subseteq S : X \in E(G) \Rightarrow Y \in E(G)$ (outcome monotonicity)

E5 $\forall s \in S \forall G_1, G_2 \subseteq N \forall X, Y \subseteq S : X \in E(G_1)$ and $Y \in E(G_2) \Rightarrow X \cap Y \in E(G_1 \cup G_2)$, where $G_1 \cap G_2 = \emptyset$ (superadditivity)

E6 $E^{nc}(\emptyset) \neq \emptyset$, where $E^{nc}(\emptyset)$ is the *non-monotonic core* of the empty coalition, namely

$$E^{nc}(\emptyset) = \{X \in E(\emptyset) : \neg\exists Y (Y \in E(\emptyset) \text{and } Y \subset X)\}$$

An effectivity function that only satisfies E1-E5 is called *playable*. On finite domains an effectivity function is playable iff it is truly playable [6], because on finite domains E6 follows from E1-E5.

A $\mathcal{CL}$ formula is interpreted in a state s in a coalition model M as follows:

$M, s \models p$ iff $p \in V(s)$

$M, s \models \neg\phi$ iff $M, s \not\models \phi$

$M, s \models (\phi_1 \wedge \phi_2)$ iff ($M, s \models \phi_1$ and $M, s \models \phi_2$)

$M, s \models [G]\phi$ iff $\phi^M \in E(s)(G)$

where $\phi^M = \{t \in S : M, t \models \phi\}$.

The axiomatisation CL of coalition logic consist of the following axioms and rules:

Prop Substitution instances of propositional tautologies

G1 $\neg[G]\bot$

G2 $[G]\top$

G3 $\neg[\emptyset]\neg\phi \to [N]\phi$

G4 $[G](\phi \wedge \psi) \to [G]\psi$

G5 $[G_1]\phi \wedge [G_2]\psi \to [G_1 \cup G_2](\phi \wedge \psi)$, if $G_1 \cap G_2 = \emptyset$

MP $\vdash_{CL} \phi, \phi \to \psi \Rightarrow \vdash_{CL} \psi$

RG $\vdash_{CL} \phi \leftrightarrow \psi \Rightarrow \vdash_{CL} [G]\phi \leftrightarrow [G]\psi$

CL is sound and complete wrt. all coalition models [17].

The following *monotonicity rule* is derivable [17], and will be useful later:

Mon $\vdash_{CL} \phi \to \psi \Rightarrow \vdash_{CL} [G]\phi \to [G]\psi$

2.2 Adding Knowledge Operators

Epistemic extensions of coalition logic were first proposed in [19][2]. They are obtained by extending the language with *epistemic operators*, and the models with *epistemic accessibility relations*.

An epistemic accessibility relation for agent i over a set of states S is a binary equivalence relation $\sim_i \subseteq S \times S$. An *epistemic coalition model*, henceforth often simply called a *model*, is a tuple

$$M = \langle S, E, \sim_1, \ldots, \sim_n, V \rangle$$

where $\langle S, E, V \rangle$ is a coalition model and $\sim_i$ is an epistemic accessibility relation over S for each agent i.

Epistemic operators come in two types: individual knowledge operators K_i, where i is an agent, and group knowledge operators C_G and D_G where G is a coalition for expressing *common knowledge* and *distributed knowledge*, respectively. Formally, the language of $\mathcal{CLCD}$ (*coalition logic with common and distributed knowledge*), is defined by extending coalition logic with all of these operators:

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid [H]\phi \mid K_i\phi \mid C_G\phi \mid D_G\phi$$

where $p \in \Theta$, $i \in N$, $H \subseteq N$ and $\emptyset \neq G \subseteq N$. When G is a coalition, we write $E_G\phi$ as a shorthand for $\bigwedge_{i \in G} K_i\phi$ (everyone in G knows ϕ).

The languages of the logics $\mathcal{CLK}$, $\mathcal{CLC}$ and $\mathcal{CLD}$ are the restrictions of this language with no C_G and no D_G operators, no D_G operators, and no C_G operators, respectively.

The interpretation of these languages in an (epistemic coalition) model M is defined by adding the following clauses to the definition for $\mathcal{CL}$:

$M, s \models K_i\phi$ iff $\forall t \in S, (s,t) \in \sim_i \Rightarrow M, t \models \phi$

$M, s \models C_G\phi$ iff $\forall t \in S, (s,t) \in (\bigcup_{i \in G} \sim_i)^* \Rightarrow M, t \models \phi$

$M, s \models D_G\phi$ iff $\forall t \in S, (s,t) \in (\bigcap_{i \in G} \sim_i) \Rightarrow M, t \models \phi$

where R^* denotes the transitive closure of the relation R. We use $\models \phi$ to denote the fact that ϕ is *valid*, i.e., that $M, s \models \phi$ for all M and states s in M.

2.2.1 Some Auxiliary Definitions

The following are some auxiliary concepts that will be useful in the following.

Intuitively, a *pseudomodel* is like a model except that distributed knowledge is "not quite" the intersection of individual knowledge. Formally, a pseudomodel is a tuple $M = (S, \{\sim_i : i \in N\}, \{R_G : \emptyset \neq G \subseteq N\}, E, V)$ where $(S, \{\sim_i : i \in N\}, E, V)$ is a model and:

- $R_G \subseteq S \times S$ is an equivalence relation for each $G \subseteq N$, $G \neq \emptyset$
- For any $i \in N$, $R_i = \sim_i$
- For any G, H, $G \subseteq H$ implies that $R_H \subseteq R_G$

The interpretation of a $\mathcal{CLCD}$ formula in a state of a pseudomodel is defined as for a model, except for the case for D_G which is interpreted by the R_G relation:

$M, s \models D_G\phi$ iff $\forall t \in S, (s,t) \in R_G \Rightarrow M, t \models \phi$

[2]In that paper for $\mathcal{ATL}$; $\mathcal{CL}$ is a fragment of $\mathcal{ATL}$.

An *epistemic model* is a model without the E function, i.e., a tuple $\langle S, \sim_1, \ldots, \sim_n, V \rangle$. An *epistemic pseudomodel* is a pseudomodel without the E function, i.e., a tuple $\langle S, \{\sim_i : i \in N\}, \{R_G : \emptyset \neq G \subseteq N\}, V \rangle$ (where R_G has the properties above). When $M = \langle S, E, \sim_1, \ldots, \sim_n, V \rangle$ is a model or $M = \langle S, \{\sim_i : i \in N\}, \{R_G : \emptyset \neq G \subseteq N\}, E, V \rangle$ is a pseudomodel, we refer to $\langle S, \sim_1, \ldots, \sim_n, V \rangle$ as M's *underlying* epistemic model.

Finally, a *playable (pseudo)model* is like a (pseudo)model, except that E is not requried to satisfy the E6 property.

We say that a formula ϕ is *satisfied in* a (playable) (pseudo)model M, if $M, s \models \phi$ for some state s in M.

3. COALITION LOGIC WITH COMMON KNOWLEDGE

In this section we consider the logic $\mathcal{CLC}$, extending coalition logic with individual knowledge operators and common knowledge. The axiomatisation CLC is the result of extending CL with the following standard axioms and rules for individual and common knowledge (see, e.g., [5]):

K $K_i(\phi \to \psi) \to (K_i\phi \to K_i\psi)$

T $K_i\phi \to \phi$

4 $K_i\phi \to K_iK_i\phi$

5 $\neg K_i\phi \to K_i\neg K_i\phi$

C $C_G\phi \to E_G(\phi \wedge C_G\phi)$

RN $\vdash_{CLC} \phi \Rightarrow \vdash_{CLC} K_i\phi$

RC $\vdash_{CLC} \phi \to E_G(\phi \wedge \psi) \Rightarrow \vdash_{CLC} \phi \to C_G\psi$

It is easy to show that CLC is sound wrt. all models.

LEMMA 1 (SOUNDNESS). *For any CLC-formula ϕ, $\vdash_{CLC} \phi \Rightarrow \models \phi$.*

Outline of completeness proof.

In the remainder of this section we show that $\mathcal{CLC}$ also is complete. Before giving all the details, we describe the outline of the proof. We first construct a canonical playable model M^c, using standard definitions of the canonical epistemic accessibility relations [9] and Pauly's definition of the canonical effectivity functions [17]. There are two potential problems with M^c: first, it is not necessarily truly playable (i.e., it is not necessarily a model), and, second, the truth lemma does not necessarily work for the case $C_G\phi$. To take care of these problems we filtrate M^c through an appropriately defined closure of a given consistent formula, to obtain a finite model M^f. This is a standard technique for dealing with transitive closure operators such as the Kleene star in $\mathcal{PDL}$ [10] and indeed common knowledge. In our case the standard technique must be extended to deal with the effectivity functions. For us the technique has the convenient side effect that playability and true playability coincides (E1-E5 implies E6) on the resulting model, since it is finite. However, it remains to be shown that filtration does not break the playability properties E1-E5, and that M^f satisfies the truth lemma for the combined (epistemic-coalitonal) language.

Completeness proof.

Theorem 1. *Any CLC-consistent formula is satisfied in some playable model.*

Proof. We define a canonical playable model $M^c = (S^c, \{\sim_i^c : i \in N\}, E^c, V^c)$ as follows:

S^c is the set of all maximally CLC consistent sets of formulas

$s \sim_i^c t$ iff $\{\psi : K_i\psi \in s\} = \{\psi : K_i\psi \in t\}$

$X \in E^c(s)(G)$ (for $G \neq N$) iff there exists ψ such that $\{t \in S^c : \psi \in t\} \subseteq X$ and $[G]\psi \in s$.

$X \in E^c(s)(N)$ iff $S^c \setminus X \notin E^c(s)(\emptyset)$.

V^c: $p \in V^c(s)$ iff $p \in s$

That $\sim_i^c$ is an equivalence relation is immediate. That $E^c(s)$ is playable (satisfies E1-E5) can be shown in exactly the same way as in the completeness proof for CL [17]. The idea behind the model construction of course is that a formula belongs to a state s in a model iff it is true there (the truth lemma). However, the canonical model is in general *not* guaranteed to satisfy every consistent formula in the $\mathcal{CLC}$ language; the case of C_G in the truth lemma does not necessarily hold. Therefore we are going to transform M^c by filtration into a finite model for a given CLC consistent formula ϕ. Note that since ϕ is consistent, it will belong to at least one s in M^c.

Let $cl(\phi)$ be the set of subformulas of ϕ closed under single negations and the condition that $C_G\psi \in cl(\phi) \Rightarrow K_iC_G\psi \in cl(\phi)$ for all $i \in G$. We are going to filtrate M^c through $cl(\phi)$. The resulting model $M^f = (S^f, \{\sim_i^f : i \in N\}, E^f, V^f)$ is constructed as follows:

S^f is $\{[s]_{cl(\phi)} : s \in S^c\}$ where $[s]_{cl(\phi)} = s \cap cl(\phi)$. We will omit the subscript $cl(\phi)$ in what follows for readability.

$[s] \sim_i^f [t]$ iff $\{\psi : K_i\psi \in [s]\} = \{\psi : K_i\psi \in [t]\}$

$V^f(s) = \{p : p \in [s]\}$. Again we will omit the subscript for readability.

$X \in E^f([s])(G)$ iff $\{s' : \phi_X \in s'\} \in E^c(s)(G)$ where $\phi_X = \vee_{[t]\in X}\phi_{[t]}$ and $\phi_{[t]}$ is a conjunction of all formulas in $[t]$.

We now prove by induction on the size of θ that for every $\theta \in cl(\phi)$, $M^f, [s] \models \theta$ iff $\theta \in [s]$.

case $\theta = p$ trivial

case booleans trivial

case $\theta = K_i\psi$ assume $M^f, [s] \not\models K_i\psi$. The latter means there is a $[s']$ such that $[s] \sim_i^f [s']$ and $M^f, [s'] \not\models \psi$. By the inductive hypothesis $\psi \notin [s']$. Since $[s']$ is deductively closed wrt $cl(\phi)$ and $K_i\psi \in cl(\phi)$, also $K_i\psi \notin [s']$. $[s] \sim_i^f [s']$ means that $[s]$ and $[s']$ contain the same K_i formulas from $cl(\phi)$, hence $K_i\psi \notin [s]$.

Assume $M^f, [s] \models K_i\psi$. Then for all $[s']$ such that $[s] \sim_i^f [s']$, $M^f, [s'] \models \psi$. This means by the IH that $\psi \in [s']$ for all $[s'] \sim_i^f [s]$. Assume by contradiction that $K_i\psi \notin [s]$. Then $\phi_{[s]}$, where $\phi_{[s]}$ is the conjunction of all formulas in $[s]$, is consistent with $\neg K_i\psi$. If we write $\langle K_i\rangle$ for the dual of the K_i modality, this is equivalent to: $\phi_{[s]} \wedge \langle K_i\rangle\neg\psi$ is consistent. By forcing choices,

$$\phi_{[s]} \wedge \langle K_i\rangle \bigvee_{\neg\psi\in[t]} \phi_{[t]}$$

is consistent. By the distributivity of $\langle K_i\rangle$ over $\vee$,

$$\bigvee_{\neg\psi\in[t]} (\phi_{[s]} \wedge \langle K_i\rangle\phi_{[t]})$$

is consistent. So for some $[t]$ with $\neg\psi \in [t]$, $\phi_{[s]} \wedge \langle K_i\rangle\phi_{[t]}$ is consistent. We claim that $[s] \sim_i^f [t]$. If this is the case, we have a contradiction, since we assumed that $\psi \in [s']$ for all $[s'] \sim_i^f [s]$.

Proof of the claim: if $\phi_{[s]} \wedge \langle K_i\rangle\phi_{[t]}$ is consistent, then $[s] \sim_i^f [t]$. Suppose not $[s] \sim_i^f [t]$, that is there is a formula χ such that $K_i\chi \in [s]$ and $\neg K_i\chi \in [t]$ or vice versa. Then we have $K_i\chi \wedge \phi_{[s]} \wedge \langle K_i\rangle(\neg K_i\chi \wedge \phi_{[t]})$ is consistent, but since K_i is an S5 modality, this is impossible. Same for the case when $\neg K_i\chi \in [s]$ and $K_i\chi \in [t]$.

case $\theta = [G]\psi$

$M^f, [s] \models [G]\psi$ iff $\psi^{M^f} \in E^f([s])(G)$ iff $\{s' : (\vee_{[t]\in\psi^{M^f}}\phi_{[t]}) \in s'\} \in E^c(s)(G)$ iff (by the IH) $\{s' : (\vee_{\psi\in[t]}\phi_{[t]}) \in s'\} \in E^c(s)(G)$ iff(*) $\{s' : \psi \in s'\} \in E^c(s)(G)$ iff(**) $[G]\psi \in s$ iff (since $[G]\psi \in cl(\phi)$) $[G]\psi \in [s]$.

Proof of (*): assume S^f contains $n+k$ states, $[t_1], \dots, [t_n]$ contain ψ and $[s_1], \dots, [s_k]$ contain $\neg\psi$. Clearly, $\phi_{[t_1]} \vee \dots \vee \phi_{[t_n]} \vee \phi_{[s_1]} \dots \vee \phi_{[s_k]}$ is provably equivalent to $\top$. Consider $\vee_{\psi\in[t]}\phi_{[t]}$. It is provably equivalent to $(\psi \wedge \phi_{[t_1]}) \vee \dots \vee (\psi \wedge \phi_{[t_n]})$. Since for every $[s_i]$ such that $\neg\psi \in [s_i]$, $(\psi \wedge \phi_{[s_i]})$ is provably equivalent to $\perp$,

$$(\psi \wedge \phi_{[t_1]}) \vee \dots \vee (\psi \wedge \phi_{[t_n]})$$

is provably equivalent to

$$(\psi\wedge\phi_{[t_1]})\vee\dots\vee(\psi\wedge\phi_{[t_n]})\vee(\psi\wedge\phi_{[s_1]})\vee\dots\vee(\psi\wedge\phi_{[s_k]})$$

which in turn is provably equivalent to

$$\psi \wedge (\phi_{[t_1]} \vee \dots \vee \phi_{[s_k]})$$

which in turn is equivalent to $\psi \wedge \top$ hence to ψ. So in M^c, $\{s' : (\vee_{\psi\in[t]}\phi_{[t]}) \in s'\} = \{s' : \psi \in s'\}$.

Proof of (**): since we defined $X \in E^c(s)(N)$ to hold iff $S^c \setminus X \notin E^c(s)(\emptyset)$, it suffices to show the case that $G \neq N$. The direction to the left is immediate: if $[G]\psi \in s$ then $\{s' \in S^c : \psi \in s'\} \in E^c(s)(G)$ by definition. For the other direction assume that $\{s' \in S^c : \psi \in s'\} \in E^c(s)(G)$, i.e., there is some γ such that $\{s' \in S^c : \gamma \in s'\} \subseteq \{s' \in S^c : \psi \in s'\}$ and $[G]\gamma \in s$. It is easy to see that $\{s' \in S^c : \gamma \in s'\} \subseteq \{s' \in S^c : \psi \in s'\}$ implies that $\vdash \gamma \to \psi$, and by the monotonicity rule it follows that $[G]\psi \in s$.

case $\theta = C_G\psi$ The proof is similar to in [20]. First we show that in M^f, if $C_G\psi \in cl(\phi)$, then $C_G\psi \in [s]$ iff every state on every $\sup_{i\in G} \sim_i^f$ path from $[s]$ contains ψ.

Suppose $C_G\psi \in [s]$. The proof is by induction on the length of the path. If the path is of 0 length, then clearly by deductive closure and by $\psi \in cl(\phi)$ we have $\psi \in [s]$. We also have $C_G\psi \in [s]$ by the assumption. IH: if $C_G\psi \in [s]$, then every state on every $\cup_{i\in G} \sim_i^f$ path of length n from $[s]$ contains ψ *and* $C_G\psi$. Inductive step: let us prove this for paths of length $n+1$. Suppose we have a path $[s] \sim_{i_1}^f [s_1] \ldots \sim_{i_n}^f [s_n] \sim_{i_{n+1}}^f [s_{n+1}]$. By the IH, $\psi, C_G\psi \in [s_n]$. Since s_n is deductively closed and $K_{i_{n+1}}C_G\psi \in cl(\phi)$, we have $K_{i_{n+1}}C_G\psi \in [s_n]$. Since $[s_n] \sim_{i_{n+1}}^f [s_{n+1}]$ and the definition of $\sim_{i_{n+1}}^f$, $C_G\psi \in [s_{n+1}]$ and hence by reflexivity $\psi \in [s_{n+1}]$.

For the other direction, suppose that every state on every $\cup_{i\in G} \sim^f$ path from $[s]$ contains ψ. Prove that $C_G\psi \in [s]$. Let $S_{G,\psi}$ be the set of all $[t]$ such that every state on every $\cup_{i\in G} \sim^f$ path from $[t]$ contains ψ. Note that each $[t]$ is/corresponds to a finite set of formulas so we can write its conjunction $\phi_{[t]}$. Consider a formula

$$\chi = \bigvee_{[t]\in S_{G,\psi}} \phi_{[t]}$$

Similarly to [20] it can be proved that $\vdash_{CLC} \phi_{[s]} \to \chi$, $\vdash_{CLC} \chi \to \psi$ and $\vdash_{CLC} \chi \to E_G\chi$. And from that follows that $\vdash_{CLC} \phi_{[s]} \to C_G\psi$ hence $C_G\psi \in [s]$.

Now we prove that $M^f, [s] \models C_G\psi$ iff $C_G\psi \in [s]$. $C_G\psi \in [s]$ iff every state on every $\cup_{i\in G} \sim_i^f$ path from $[s]$ contains ψ iff for every $[t]$ reachable from $[s]$ by a $\cup_{i\in G} \sim_i^f$ path, $M^f, [t] \models \psi$ iff $M^f, [s] \models C_G\psi$.

□

It is obvious that in M^f, $\sim_i$ are equivalence relations. So what remains to be proved is that E^f satisfies E1-E6. Since S^f is finite, it suffices to show E1-E5, which for finite sets of states entail E6.

Proposition 1. *M^f satisfies E1-E5.*

Proof.

E1 Note that $\phi_\emptyset$ is the empty disjunction, $\bot$.
$\emptyset \in E^f([s])(G)$ iff (by definition of E^f) $\{s' : \bot \in s'\} \in E^c(s)(G)$ iff $\emptyset \in E^c(s)(G)$. Since E^c satisfies E1, $\emptyset \notin E^f([s])(G)$.

E2 $S^f \in E^f([s])(G)$ iff $\{s' : \bigvee_{[t]\in S^f} \in s'\} \in E^c(s)(G)$ iff $S^c \in E^c(s)(G)$. Since E^c satisfies $E2$, $S^f \in E^f([s])(G)$.

E3 Let $\overline{X} \notin E^f([s])(\emptyset)$. Then $\{s' : \phi_{\overline{X}} \in s'\} \notin E^c(s)(\emptyset)$. Note that $\{s' : \phi_{\overline{X}} \in s'\}$ is the complement of $\{s' : \phi_X \in s'\}$, since $\phi_{\overline{X}} = \neg\phi_X$. Since E^c satisfies E3, this means that $\{s' : \phi_X \in s'\} \in E^c(s)(N)$. Hence $X \in E^f([s])(N)$.

E4 Let $X \subseteq Y \subseteq S^f$ and $X \in E^f([s])(G)$. Clearly $\vdash_{CLC} \phi_X \to \phi_Y$. Hence $\{s' : \phi_X \in s'\} \subseteq \{s' : \phi_Y \in s'\}$. Since $X \in E^f([s])(G)$, we have $\{s' : \phi_X \in s'\} \in E^c(s)(G)$. Since E^c satisfies E4, $\{s' : \phi_Y \in s'\} \in E^c(s)(G)$ so $Y \in E^f([s])(G)$.

E5 Let $X \in E^f([s])(G_1)$ and $Y \in E^f([s])(G_2)$ and $G_1 \cap G_2 = \emptyset$. So $\{s' : \phi_X \in s'\} \in E^c(s)(G_1)$ and $\{s' : \phi_Y \in s'\} \in E^c(s)(G_2)$ and since E^c satisfies E5, $\{s' : \phi_X \in s'\} \cap \{s' : \phi_Y \in s'\} \in E^c(s)(G_2)$. Note that $\{s' : \phi_X \in s'\} \cap \{s' : \phi_Y \in s'\} = \{s' : (\vee_{[t]\in X}\phi_{[t]}) \in s'$ and $(\vee_{[t]\in Y}\phi_{[t]}) \in s'\}$ which is in turn the same as

$$\{s' : (\vee_{[t]\in X\cap Y}\phi_{[t]}) \in s'\}$$

since $\{s' : (\vee_{[t]\in X\cap Y}\phi_{[t]}) \in s'\} \in E^c(s)(G_2)$, $X \cap Y \in E^f([s])(G_1)$.

□

Corollary 1. *For any CLC-formula ϕ, $\vdash_{CLC} \phi$ iff $\models \phi$.*

4. EPISTEMIC COALITION LOGIC WITH DISTRIBUTED KNOWLEDGE

In this section we consider the logic $\mathcal{CLD}$, extending coalition logic with individual knowledge operators and distributed knowledge.

The axiomatisation CLD is obtained by extending CL with the following standard axioms and rules for individual and distributed knowledge (see, e.g., [5]):

K $K_i(\phi \to \psi) \to (K_i\phi \to K_i\psi)$

T $K_i\phi \to \phi$

4 $K_i\phi \to K_iK_i\phi$

5 $\neg K_i\phi \to K_i\neg K_i\phi$

RN $\vdash_{CLD} \phi \Rightarrow \vdash_{CLD} K_i\phi$

DK $D_G(\phi \to \psi) \to (D_G\phi \to D_G\psi)$

DT $D_G\phi \to \phi$

D4 $D_G\phi \to D_GD_G\phi$

D5 $\neg D_G\phi \to D_G\neg D_G\phi$

D1 $K_i\phi \leftrightarrow D_i\phi$

D2 $D_G\phi \to D_H\phi$, if $G \subseteq H$

As usual, soundness can easily be shown.

Lemma 2 (Soundness). *For any CLD-formula ϕ, $\vdash_{CLD} \phi \Rightarrow \models \phi$.*

Outline of completeness proof.

In the remainder of this section we show that CLD also is complete. An outline of the proof is as follows. As in the case of CLC, we start with the canonical model construction. However, rather than constructing a playable model, we construct a playable *pseudo*model M^c. The truth lemma for the combined epistemic-coalitional language holds for M^c, but the relations interpreting distributed knowledge are not necessarily the intersections of the individual epistemic accessibility relations. The idea is to transform M^c into a proper model, which has the E1-E6 propeties, without breaking the truth lemma. This is done in two additional steps. First, M^c is transformed into a finite pseudomodel M^f, exactly like in the case of CLC. The transformation preserves satisfaction, as well as the playability properties (and E6 follows from finiteness). Using pseudomodels that are then transformed into proper models is a common way to deal with intersection in general and distributed knowledge in particular [22]. We can in fact now make directly use of an existing completeness result and proof for epistemic logic with distributed knowledge [5], by taking the (finite) epistemic pseudomodel underlying M^f and transform it into a proper (not necessarily finite) epistemic model which is used as the underying epistemic model of the final model M'. It remains to be shown that the transformation did not break the true playability properties, nor satisfaction of formulae in the closure.

Completeness proof.

For a set of formulae s, let $K_a s = \{K_a\phi : K_a\phi \in s\}$ and $D_G s = \{D_G\phi : D_G\phi \in s\}$.

Definition 1 (Canonical Playable Pseudomodel). *The canonical playable pseudomodel $M^c = (S^c, \{\sim_i^c : i \in N\}, \{R_G^c : \emptyset \neq G \subseteq N\}, E^c, V^c)$ for $\mathcal{CLD}$ is defined as follows:*

- *S^c is the set of maximal consistent sets.*
- *$s \sim_i^c t$ iff $K_i s = K_i t$*
- *$sR_G t$ iff $D_H s = D_H t$ whenever $H \subseteq G$*
- *$X \in E^c(s)(G)$ (for $G \neq N$) iff there exists ψ such that $\{t \in S^c : \psi \in t\} \subseteq X$ and $[G]\psi \in s$.*
- *$X \in E^c(s)(N)$ iff $S^c \setminus X \notin E^c(s)(\emptyset)$.*
- *$V^c(s) = \{p : p \in s\}$*

Lemma 3 (Pseudo Truth Lemma). *$M^c, s \models \phi \Leftrightarrow \phi \in s$.*

Proof. The proof is by induction on ϕ. The epistemic cases are exactly as for standard normal modal logic. The case for coalition operators is exactly as in [17]. □

It is easy to check that $\sim_i^c$ are equivalence relations and E1-E5 hold for E^c.

Lemma 4 (Finite Pseudomodel). *Every CLD-consistent formula ϕ is satisfied in a finite pseudomodel where E1-E6 hold.*

Proof. The proof is exactly as in Theorem 1, namely the construction of M^f, but starting with a Canonical Playable Pseudomodel rather than Canonical Playable Model; the definition of M^c contains the clause

$$\Gamma R_G \Delta \text{ iff } \forall H \subseteq G \{\psi : D_H\psi \in \Gamma\} = \{\psi : D_H\psi \in \Delta\}$$

We add the following condition to the closure: $D_i\psi \in cl(\phi)$ iff $K_i\psi \in cl(\phi)$.

We define M^f to be a pseudomodel instead of a model, by adding the clause:

$$[s]R_G^f[s'] \text{ iff } \forall H \subseteq G \{\psi : D_H\psi \in [s]\} = \{\psi : D_H\psi \in [s']\}$$

We show that M^f is indeed a pseudomodel:

- $R_i^f = \sim_i^f$: this follows from the fact that $K_i\phi \in [s]$ iff $D_i\phi \in [s]$ for any i, ϕ and s, which holds because of the $K_i\phi \to D_i\phi$ axiom and the new closure condition above.
- $G \subseteq H \Rightarrow R_H^f \subseteq R_G^f$: this holds by definition.

We add a case for $\theta = D_G\psi$ to the inductive proof. This case is proven in exactly the same way as the $\theta = K_i\psi$ case: the definitions of $\sim_i^f$ and R_G^f are of exactly the same form (in particular, R_G^f is also an $S5$ modality). The proof that E1-E6 hold in the resulting pseudomodel is the same as in the proof of Theorem 1 for E^f. □

We are now going to transform the pseudomodel into a proper model; it is a well-known technique for dealing with distributed knowledge. In fact, we can make direct use of a corresponding existing result for epistemic logic with distributed knowledge, and extend it with the coalition operators/effectivity functions. We here give the more general result for the language with also common knowledge, which will be useful later.

Theorem 2 ([5]). *If $M_p = (S, \{\sim_i : i \in N\}, \{R_G : \emptyset \neq G \subseteq N\}, V)$ is an epistemic pseudomodel, then there is an epistemic model $M'_p = (S', \{\sim'_i : i \in N\}, V')$ and a surjective (onto) function $\mathbf{f} : S' \to S$ such that for every $s' \in S'$ and formula $\phi \in \mathcal{ELCD}$, $M_p, \mathbf{f}(s') \models \phi$ iff $M'_p, s' \models \phi$.*

Proof. This result is directly obtained from the completeness proof for $\mathcal{ELCD}$ sketched in [5, p. 70]. For a more detailed proof (for a more general language), see [22, Theorem 9]. □

Theorem 3. *If a formula is satisfied in some finite pseudomodel, then it is satisfied in some model.*

Proof. Let $M = (S, \{\sim_i : i \in N\}, \{R_G : \emptyset \neq G \subseteq N\}, E, V)$ be a finite pseudomodel such that $M, s \models \phi$. Let $M_p = (S, \{\sim_i : i \in N\}, \{R_G : \emptyset \neq G \subseteq N\}, V)$ be the epistemic pseudomodel underlying M, and let $M'_p = (S', \{\sim'_i : i \in N\}, V')$ and $\mathbf{f} : S' \to S$ be as in Theorem 2. Let $\mathbf{f}^{-1}(X) = \{s' \in S' : \mathbf{f}(s') \in X\}$ for any set $X \subseteq S$. Finally, let $M' = (S', \{\sim'_i : i \in N\}, E', V')$ where E' is defined as follows:

- For $G \neq N$: $Y \in E'(u)(G) \Leftrightarrow \exists X \subseteq S, (Y \supseteq \mathbf{f}^{-1}(X)$ and $X \in E(\mathbf{f}(u))(G))$
- for $G = N$: $Y \in E'(u)(G) \Leftrightarrow \overline{Y} \notin E'(u)(\emptyset)$

Two things must be shown: that M' is a proper model, and that it satisfies ϕ.

Since M'_p is an epistemic model, to show that M' is a model all that remains to be shown is that E' is truly playable. We now show that that follows from true playability of E.

E1 Note that $\mathbf{f}^{-1}(X) = \emptyset$ iff $X = \emptyset$.

For $G \neq N$, $\emptyset \in E'(u)(G)$ iff (by definition of E') $\exists X \subseteq S, (\emptyset \supseteq \mathbf{f}^{-1}(X)$and $X \in E(\mathbf{f}(u))(G))$ iff $\emptyset \in E(\mathbf{f}(u))(G))$ which is impossible since M satisfies E1. Note that in particular this proves $\emptyset \notin E'(u)(\emptyset)$, which we will use in the E2 case below.

For $G = N$, $\emptyset \in E'(u)(G)$ iff $S' \notin E'(u)(\emptyset)$ and we'll see that this is impossible in the E2 case below.

E2 Note that $\mathbf{f}^{-1}(S) = S'$.

For $G \neq N$, $S' \in E'(u)(G)$ iff (by definition of E') $\exists X \subseteq S$, $(S' \supseteq \mathbf{f}^{-1}(X)$and $X \in E(\mathbf{f}(u))(G))$, and by taking $X = S$ we get that $S' \in E'(u)(G)$ holds since $S' \supseteq \mathbf{f}^{-1}(S)$ and $S \in E(\mathbf{f}(u))(G)$. Note that in particular this proves $S' \in E'(u)(\emptyset)$, which we needed in the E1 case above.

For $G = N$, $S' \in E'(u)(G)$ iff $\emptyset \notin E'(u)(\emptyset)$ and this was proved in the E1 case above.

E3 $\forall u \in S' \forall Y \subseteq S'\ \overline{Y} \notin E'(u)(\emptyset) \Rightarrow Y \in E'(u)(N)$ follows immediately from the definition for $E'(u)(N)$.

E4 E' is monotonic by definition for $G \neq N$.

For N, assume $X \subseteq Y$ and $X \in E'(u)(N)$. Then $\overline{X} \notin E'(u)(\emptyset)$. Since we already know that E' is monotonic for $G = \emptyset$ and $\overline{Y} \subseteq \overline{X}$, $\overline{Y} \notin E'(u)(\emptyset)$. So $Y \in E'(u)(N)$.

E5 Let $u \in S'$, $f(u) = s$, $G_1, G_2 \subseteq N$ such that $G_1 \cap G_2 = \emptyset$, $X', Y' \subseteq S'$, $X' \in E'(u)(G_1)$ and $Y' \in E'(u)(G_2)$. We must show that $X' \cap Y' \in E'(u)(G_1 \cup G_2)$. We reason by cases for G_1 and G_2.

First consider the case that $G_1 \cup G_2 \neq N$. We must show that there is a Z such that $\mathbf{f}^{-1}(Z) \subseteq X' \cap Y'$ and $Z \in E(s)(G_1 \cup G_2)$. We have that there are X, Y such that $\mathbf{f}^{-1}(X) \subseteq X'$ and $X \in E(s)(G_1)$ and $\mathbf{f}^{-1}(Y) \subseteq Y'$ and $Y \in E(s)(G_2)$. Take $Z = X \cap Y$. It is easy to see that $\mathbf{f}^{-1}(X \cap Y) = \mathbf{f}^{-1}(X) \cap \mathbf{f}^{-1}(Y)$ (from the defintion of $\mathbf{f}^{-1}(\cdot)$), and we thus get that $\mathbf{f}^{-1}(Z) = \mathbf{f}^{-1}(X) \cap \mathbf{f}^{-1}(Y) \subseteq X' \cap Y'$. From $X \in E(s)(G_1)$ and $Y \in E(s)(G_2)$ and superadditivity of E we get that $Z \in\in E(s)(G_1 \cup G_2)$.

Second consider the case that $G_1 = N$ or $G_2 = N$. Wlog. assume the former. That implies that $G_2 = \emptyset$. We must show that $X' \cap Y' \in E'(u)(N)$, i.e., that $\overline{X' \cap Y'} \notin E'(u)(\emptyset)$. Assume otherwise, i.e., that $\overline{X' \cap Y'} \in E'(u)(\emptyset)$, in other words that $\overline{X'} \cup \overline{Y'} \in E'(u)(\emptyset)$. As $G_2 = \emptyset$ we also have that $Y' \in E'(u)(\emptyset)$, and by E5 for E' for the case that $G_1 = G_2 = \emptyset \neq N$ (proven above) we get that $(\overline{X'} \cup \overline{Y'}) \cap Y' \in E'(u)(\emptyset)$. I.e., $\overline{X'} \cap Y' \in E'(u)(\emptyset)$. By E4 for E' (proven above), we get that $\overline{X'} \in E'(u)(\emptyset)$. But that contradicts the fact that $X' \in E(u)(G_1)$ with $G_1 = N$.

Finally, consider the case that $G_1 \cup G_2 = N$ and $G_1 \neq N$ and $G_2 \neq N$. We must show that $X' \cap Y' \in E'(u)(N)$, i.e., that $\overline{X' \cap Y'} \notin E'(u)(\emptyset)$, i.e., that there does not exist a Z such that $\mathbf{f}^{-1}(Z) \subseteq \overline{X' \cap Y'}$ and $Z \in E(s)(\emptyset)$. Assume otherwise, that such a Z exists. Let X, Y be such that

$$\begin{array}{l}\mathbf{f}^{-1}(X) \subseteq X' \text{ and } X \in E(s)(G_1)\\ \mathbf{f}^{-1}(Y) \subseteq Y' \text{ and } Y \in E(s)(G_1)\end{array}$$

which exist because $X' \in E'(u)(G_1)$ and $Y' \in E'(u)(G_2)$. From superadditivity of E we get that

$$X \cap Y \in E(s)(N) \tag{1}$$

It follows that

$$\overline{X \cap Y} \notin E(s)(\emptyset) \tag{2}$$

because otherwise $\emptyset = (X \cap Y) \cap (\overline{X \cap Y}) \in E(s)(N)$ by E5 for E, which contradicts E1 for E. We furthermore have that

$$\begin{array}{l}\overline{X'} \subseteq \overline{\mathbf{f}^{-1}(X)} \subseteq \mathbf{f}^{-1}(\overline{X})\\ \overline{Y'} \subseteq \overline{\mathbf{f}^{-1}(Y)} \subseteq \mathbf{f}^{-1}(\overline{Y})\end{array} \tag{3}$$

which follow immediately from the facts that $\mathbf{f}^{-1}(X) \subseteq X'$ and $\mathbf{f}^{-1}(Y) \subseteq Y'$ and the definition of $\mathbf{f}^{-1}(\cdot)$. From (3) it follows that

$$\overline{X'} \cup \overline{Y'} \subseteq \mathbf{f}^{-1}(\overline{X} \cup \overline{Y}) \tag{4}$$

From (4) and the assumption that $Z \in E(s)(\emptyset)$ we get that $\mathbf{f}^{-1}(Z) \subseteq \mathbf{f}^{-1}(\overline{X} \cup \overline{Y})$, and it follows, by surjectivity of $\mathbf{f}$, that

$$Z \subseteq \overline{X \cap Y} \tag{5}$$

By (5) and the assumption that $Z \in E(s)(\emptyset)$ we get that $\overline{X \cap Y} \in E(s)(\emptyset)$. But this contradicts (2).

E6 We must show that $E'^{nc}(u)(\emptyset) \neq \emptyset$, for any u. Let $s = f(u)$, and let $X \in E^{nc}(s)(\emptyset)$ (exists because of E6 for E). We show that $\mathbf{f}^{-1}(X) \in E'^{nc}(u)(\emptyset)$. First, we have that $\mathbf{f}^{-1}(X) \in E'(u)(\emptyset)$; this follows from the fact that $X \in E(s)(\emptyset)$ and the definition of E'. Second, assume, towards a contradiction, that there exists a $Y \subset \mathbf{f}^{-1}(X)$ such that $Y \in E'(u)(\emptyset)$. By the definition of E', this means that there is a Z such that $\mathbf{f}^{-1}(Z) \subseteq Y$ and $Z \in E(s)(\emptyset)$. Since $Y \subset \mathbf{f}^{-1}(X)$ and $\mathbf{f}^{-1}(Z) \subseteq Y$ it follows that $\mathbf{f}^{-1}(Z) \subset \mathbf{f}^{-1}(X)$. It is easy to see (from surjectivity of $\mathbf{f}$) that it follows that $Z \subset X$, and this contradicts the assumption that $Z \in E(s)(\emptyset)$ and $X \in E^{nc}(s)(\emptyset)$.

In order to show that M' satisfies ϕ, we show that $M, \mathbf{f}(u) \models \gamma$ iff $M', u \models \gamma$ for any $u \in S'$ and any γ, by induction in γ. All cases except $\gamma = [G]\psi$ are exactly as in the proof of Theorem 2.

For the case that $\gamma = [G]\psi$, the inductive hypothesis is that for all proper subformulae χ of $[G]\psi$, and any v, $M, \mathbf{f}(v) \models \chi$ iff $M', v \models \chi$. We can state this as $\{v : M', v \models \chi\} = \mathbf{f}^{-1}(\chi^M)$, or $\chi^{M'} = \mathbf{f}^{-1}(\chi^M)$.

First consider the case that $G \neq N$. Let $\mathbf{f}(u) = s$. $M', u \models [G]\psi$ iff $\psi^{M'} \in E'(u)(G)$ iff there is an X such that $\mathbf{f}^{-1}(X) \subseteq \psi^{M'}$ and $X \in E(s)(G)$. This holds iff $\psi^M \in E(s)(G)$ iff $M, s \models [G]\psi$. For the implication to the left take $X = \psi^M$; for the implication to the right observe that $\mathbf{f}^{-1}(X) \subseteq \mathbf{f}^{-1}(\psi^M)$ implies that $X \subseteq \psi^M$, and $\psi^M \in E(s)(G)$ follows from $X \in E(s)(G)$ by outcome monotonicity of E.

Second consider the case that $G = N$. $M, s \models [N]\psi$ iff $\psi^M \in E(s)(N)$ iff (*) $\neg\psi^M \notin E(s)(\emptyset)$ iff (as above) $\neg\psi^{M'} \notin E'(u)(\emptyset)$ iff $M', u \models [N]\psi$. (*): one direction E3, the other direction E5 and E1. $\square$

Corollary 2. *For any CLD-formula ϕ, $\vdash_{CLD} \phi$ iff $\models \phi$.*

5. EPISTEMIC COALITION LOGIC WITH BOTH COMMON AND DISTRIBUTED KNOWLEDGE

In this section we consider the logic $\mathcal{CLCD}$, extending coalition logic with operators for individual knowledge, common knowledge and distributed knowledge.

The axiomatisation $CLCD$ is obtained by extending CL with the axioms and rules of CLC and CLD.

Lemma 5 (Soundness). *For any CLCD-formula ϕ, $\vdash_{CLCD} \phi \Rightarrow \models \phi$.*

Completeness can in fact be shown in exactly the same was as for CLD, except that there is an extra clause for $C_G\phi$ in the proof of satisfaction which is taken care of in the same way as in the proof for CLC.

Theorem 4. *Any CLCD-consistent formula is satisfied in some finite pseudomodel.*

Proof. The proof is identical to the proof of Lemma 4, starting with the canonical playable pseudomodel, with the addition of the inductive clause $\theta = C_G\psi$ as in the proof of Theorem 1. $\square$

We can now use the same approach as in the case of $\mathcal{CLD}$.

Theorem 5. *If a $\mathcal{CLCD}$ formula is satisfied in some finite pseudomodel, it is satisfied in some model.*

Proof. The proof goes exactly like the proof of Theorem 3, using Theorem 2. The definition of the model M' is identical to the definition in Theorem 3, as is the proof that it is a proper model. For the last part of the proof, i.e., showing that M' satisfies ϕ, note that the last clause in Theorem 2 holds for epistemic logic with both distributed and common knowledge. Thus, the proof is completed by only adding the inductive clause for $[G]\phi$, which is done in exactly the same way as in Theorem 3. □

Corollary 3. *For any CLCD-formula ϕ, $\vdash_{CLCD} \phi$ iff $\models \phi$.*

6. COMPUTATIONAL COMPLEXITY

The following complexity result is an easy consequence of the known results for other logics:

Theorem 6. *The satisfiability problem for CLC and for CLCD is EXPTIME-complete.*

Proof. EXPTIME-hardness follows from EXPTIME-hardness of $S5_n + C$ ([9]). EXPTIME upper bound follows from the upper bound for ATEL ([21]). □

Theorem 7. *The satisfiability problem for CLD is PSPACE-complete.*

Proof. PSPACE-hardness follows from PSPACE-hardness of $S5_n$ [9] and also from PSPACE-hardness of CL [17].

The PSPACE upper bound can be obtained by combining the tableaux algorithm for $S5_n + D$ given in [9] with the algorithm in [17] which checks the satisfiability of a finite set of coalition logic formulas. The two algorithms need to call each other recursively. In addition, since [9] only has the D operator for the grand coalition, the rule for producing witnesses for the $\neg D_G\phi$ formulae has to be modified as in the tableaux algorithm for $K_\omega^{\cup,\cap}$ [15]. □

7. CONCLUSIONS

This papers settles several hitherto unsolved problems. It proves completeness of coalition logic extended with different combinations of group knowledge operators. The axioms for the epistemic modalities are standard in epistemic logic, but the completeness proofs require non-trivial combinations of techniques. The proofs are given in detail, and can be used and extended in future work. The paper furthermore completely characterises the computational complexity of the considered logics. They are all decidable. We can conclude that adding coalition operators to epistemic logic comes "for free" without changing the complexity of the satisfiability problem: the extension of epistemic logic with distributed and common knowledge with coalition operators remains EXPTIME-complete, the extension of epistemic logic with only distributed knowledge with coalition operators remains PSPACE-complete.

8. REFERENCES

[1] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49:672–713, 2002.

[2] N. Belnap and M. Perloff. Seeing to it that: a canonical form for agentives. *Theoria*, 54:175–199, 1988.

[3] J. Broersen. A complete stit logic for knowledge and action, and some of its applications. In *Proceedings of DALT 2008*, volume 5397 of *LNCS*, pages 47–59, 2009.

[4] J. Broersen, A. Herzig, and N. Troquard. A normal simulation of coalition logic and an epistemic extension. In *Proc. of TARK 2007*, pp. 92–101, 2007.

[5] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. The MIT Press, 1995.

[6] V. Goranko, W. Jamroga, and P. Turrini. Strategic games and truly playable effectivity functions. In *Proceedings of AAMAS 2011*, pages 727–734, 2011.

[7] V. Goranko, W. Jamroga, and G. van Drimmelen. Axiomatic systems for alternating time temporal epistemic logics (extended abstract). 2004.

[8] V. Goranko and G. van Drimmelen. Complete axiomatization and decidability of alternating-time temporal logic. *Theoretical Computer Science*, 353(1):93–117, 2006.

[9] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(2):319–379, 1992.

[10] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.

[11] A. Herzig and N. Troquard. Knowing how to play: Uniform choices in logics of agency. In *Proceedings of AAMAS 2006*, pages 209–216, 2006.

[12] A. H. J. Broersen and N. Troquard. What groups do, can do, and know they can do: an analysis in normal modal logics. *J. of Applied Non-Classical Logic*, 19(3):261–289, 2009.

[13] W. Jamroga and T. Ågotnes. Constructive knowledge: what agents can achieve under imperfect information. *J. of Applied Non-Classical Logics*, 17:423–475, 2007.

[14] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundam. Informaticae*, 63:185–219, 2004.

[15] C. Lutz and U. Sattler. The complexity of reasoning with boolean modal logics. In *Advances in Modal Logic*, volume 3, pages 329–348. World Scientific, 2002.

[16] J.-J. C. Meyer and W. van der Hoek. *Epistemic Logic for AI and Computer Science*. Cambridge University Press: Cambridge, England, 1995.

[17] M. Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12(1):149–166, 2002.

[18] P. Y. Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2), 2004.

[19] W. van der Hoek and M. Wooldridge. Cooperation, knowledge and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75:125–157, 2003.

[20] H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*, Springer, 2007.

[21] D. Walther. ATEL with common and distributed knowledge is ExpTime-complete. In *Proc. of M4M-4*, 2005.

[22] Y. Wang and T. Ågotnes. Public announcement logic with distributed knowledge. In *Proceedings of LORI 2011*, volume 6953 of *LNCS*. Springer, 2011.

Group Synthesis for Parametric Temporal-Epistemic Logic

Andrew V. Jones
Department of Computing
Imperial College London
andrew.jones@ic.ac.uk

Michał Knapik
ICS PAS
mknapik@ipipian.waw.pl

Alessio Lomuscio
Department of Computing
Imperial College London
a.lomuscio@ic.ac.uk

Wojciech Penczek
ICS PAS and
UPH Siedlce
penczek@ipipan.waw.pl

ABSTRACT

We investigate parameter synthesis in the context of temporal-epistemic logic. We introduce CTLPK, a parametric extension to the branching time temporal-epistemic logic CTLK with free variables representing groups of agents. We give algorithms for automatically synthesising the groups of agents that make a given parametric formula satisfied. We discuss an implementation of the technique on top of the open-source model checker MCMAS and demonstrate its attractiveness by reporting the experimental results obtained.

Categories and Subject Descriptors

D.2.4 [**Software/Program Verification**]: Model Checking

General Terms

Verification

Keywords

Temporal-Epistemic Logic, Model Checking

1. INTRODUCTION

Multi-agent systems (MAS) are distributed systems in which components, or agents, interact with one another trying to reach private or common goals. One of the recent topics of interest in this area is the issue of verification and validation of MAS, i.e., how to ascertain whether a given MAS satisfies certain specifications of interest. In this context a number of model checkers [5, 7, 9] have been developed to verify logics for MAS, including epistemic, deontic, and strategic logics.

Of particular interest to the community is work on automated model checking tailored to temporal-epistemic specifications. In this line specifications of MAS are defined on temporal languages augmented with modalities to reason about the knowledge of the agents in the system. As an example of this, most coordination protocols require common knowledge to be obtained within the group of agents before the protocol can be executed by the MAS [4]. Common knowledge (and other group modalities such as distributed knowledge [4]) are expressed in temporal-epistemic logic by using indices representing the groups they refer to. Model checkers such as MCMAS [9] and MCK [5] already support specifications with group operators including common knowledge and can be used to verify such properties in a given system.

There are scenarios, however, when checking knowledge for a specific group of agents is not sufficient. For example, in a MAS that implements distributed diagnosis we, as specifiers, would actually like to know *which groups* in the system obtain distributed knowledge of a particular fault in the system. Moreover, even if we have an intuition as to whether a particular group reaches common or distributed knowledge of a particular property, it is of interest to ascertain whether there is a maximal or minimal group that obtains this so that, for instance, we can minimise the number of agents involved in a coordination protocol.

If the set of agents is finite, we can solve this problem by repeatedly querying a model checker with all the possible instantiations of the specification for all possible valuations. However, if this set is large, its power set will not be of a trivial size, resulting in a large number of checks. If the specification of interest involves several, not necessarily equal groups, the number of instantiations grows exponentially. In symbolic model checking the bottleneck is normally the computation of the set of reachable states, but if the number of formulae to be checked is sufficiently high, the verification for the formulae can become the most time consuming operation.

The aim of this paper is to explore an alternative, potentially more efficient technique to identify (or *synthesise*) the groups of agents for which a given temporal-epistemic specification holds. We call this *parametric model checking for temporal-epistemic logic* due to its clear correspondence to parametric model checking for temporal specifications [8, 13], where temporal intervals are synthesised. Concisely, in the approach presented the groups under synthesis are treated as variables ranging over subsets of the set of all agents. The model checking algorithm we put forward returns only the subsets validating a given formula.

The rest of the paper is as follows. The syntax and semantics of CTLPK is introduced in the next section. Section 3 presents the synthesis algorithms for the verification of the parametric formulae. In Section 4 we use an experimental implementation to demonstrate results for parameter synthesis for the dining cryptographers protocol and diagnosability properties for a commonly used network protocol. Finally, in Section 5, we conclude.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

2. THE LOGIC CTLPK

In this section we introduce Parametric Computation Tree Logic with Knowledge (CTLPK, for short), a branching time temporal-epistemic logic that includes free variables as parameters for the group modalities. Intuitively, any CTLPK formula ϕ represents the set of formulae that can be constructed from ϕ by instantiating the parameters in ϕ with any non-empty set of agents in the group considered.

Definition 1 CTLPK syntax. Let $\mathcal{PV}$ be a set of *propositions*, *Groups* be a finite set of *group variables*, and $Agents = \{1, \ldots, n\}$ be a finite set of *agents*. The grammar of CTLPK in BNF is given below.

$$\begin{array}{rcl} \phi & ::= & p \mid \neg\phi \mid \phi \vee \phi \mid EX\phi \mid EG\phi \mid E\phi U\psi \mid \\ & & K_i\phi \mid E_\Gamma\phi \mid D_\Gamma\phi \mid C_\Gamma\phi \mid K_Y\phi \mid E_Y\phi \mid D_Y\phi \end{array}$$

where $p \in \mathcal{PV}$, $i \in Agents$, $\Gamma \subseteq Agents$, $\Gamma \neq \emptyset$ and $Y \in Groups$.

A formula containing at least one group variable is said to be *parametric* while a formula with no group variable is called *ground*. Notice that the set of the ground formulae defines CTLK [11].

Recall that EX, EG, EU are temporal modalities, where E stands for "there exists a path", and X, G and U respectively mean "at the next state", "for all successor states" and "until". The epistemic modalities K_i, E_Γ, D_Γ, C_Γ are interpreted as follows: $K_i\phi$ stands for "agent i knows ϕ"; E_Γ is read as "everyone in group Γ knows ϕ"; and $D_\Gamma\phi$ ($C_\Gamma\phi$) stands for "the group Γ has distributed (common, respectively) knowledge of ϕ". Let $AF\phi \overset{def}{=} \neg EG\neg\phi$ be a derived temporal modality. The formula $AF\phi$ expresses: "for all paths eventually ϕ holds".

As an example, consider the formula $\phi = AF(D_Y fault)$, where Y is a group variable, and $Agents = \{a, b\}$. The formula ϕ represents the set of ground formulae $\{AF(D_{\{a\}}fault)$, $AF(D_{\{b\}}fault)$, $AF(D_{\{a,b\}}fault)\}$. The parametric formula above states that "for all paths eventually the agents in Y will have distributed knowledge of *fault*". Since ϕ contains a free variable, similar to first-order logic, an interpretation for ϕ in a model is an assignment υ from the variable Y to concrete instances in $2^{\{a,b\}} \setminus \{\emptyset\}$.

Given that variables in CTLPK represent groups, a natural question to ask is *"which groups satisfy the temporal-epistemic specification ϕ on a given model M?"*. In the following we provide an efficient method for calculating all interpretations υ, such that all ground formulae constructed from ϕ by replacing the variable Y with $\Gamma \in \upsilon(Y)$ are satisfied in M.

Before we do so, we provide the semantics for CTLPK in terms of the interpreted systems formalism [4], a standard semantics for epistemic logic. We assume that each agent i in *Agents* is defined by means of a set of local states L_i, actions Act_i and a protocol $P_i : L_i \to 2^{Act_i}$. The environment is analogously defined by L_e, Act_e, and P_e. The set of global states $G \subseteq L_1 \times \cdots \times L_n \times L_e$ is a subset of the Cartesian product of the local states for the agents and the environment. The transitions are defined locally from local states on joint actions by considering evolution functions $\tau_i : L_i \times Act_1 \times \cdots \times Act_n \times Act_e \to L_i$. We refer to [4] for more details.

Definition 2 Interpreted Systems. Given a set of agents *Agents*, an *interpreted system* (or *model*) M is a tuple $M = (G, g^0, T, \sim_1, \ldots, \sim_n)$ such that:

- $G \subseteq L_1 \times \cdots \times L_n \times L_e$ is the set of *reachable* global states for the system, where $g^0 \in G$.
- The transition relation $T \subseteq G \times G$ is defined by $(g, g') \in T$ if there exists $(act_1, \ldots, act_n, act_e) \in Act_i \times \cdots \times Act_n \times Act_e$ such that $\tau_i(l_i(g), act_1, \ldots, act_n, act_e) = l_i(g')$ for all $i \in Agents$ and $\tau_e(l_e(g), act_1, \ldots, act_n, act_e) = l_e(g')$, where $l_i(g)$ returns the local state of agent i in the global state g. The actions $act_1, \ldots, act_n, act_e$ are all consistent with their respective protocols, i.e., $act_i \in P_i(l_i(g))$ and analogously for the environment.
- For any $i \in Agents$ the relation $\sim_i \subseteq G \times G$ is an epistemic accessibility relation such that $g \sim_i g'$ iff $l_i(g) = l_i(g')$, where $l_i(g)$ is as above.
- The function $\mathcal{L} : G \to 2^{\mathcal{PV}}$ is an interpretation for a set of the propositions $\mathcal{PV}$.

Given a concrete group Γ of agents, we introduce three group accessibility relations as follows:

$$\sim_\Gamma^E = \bigcup_{i\in\Gamma} \sim_i, \quad \sim_\Gamma^D = \bigcap_{i\in\Gamma} \sim_i, \quad \sim_\Gamma^C = \left(\sim_\Gamma^E\right)^+,$$

where $^+$ denotes the transitive closure. These relations are used to interpret the ground group modalities.

We now introduce the notion of a *path* as a sequence $\pi = (g_0, g_1, \ldots)$ such that $g_i \in G$ and $(g_i, g_{i+1}) \in T$ for all $i \geq 0$. We denote $\pi(j) = g_j$ for all $j \geq 0$.

In the definition of the semantics of the CTLPK formulae, we use a valuation of the group variables $\upsilon : Groups \to 2^{Agents} \setminus \{\emptyset\}$. The set of all the valuations of the group variables is denoted by *GroupVals*. Formally

$$GroupVals = \left(2^{Agents} \setminus \{\emptyset\}\right)^{Groups}.$$

Now we are in the position to introduce the semantics of CTLPK. If ϕ is a formula of CTLPK, then by $M, g \models_\upsilon \phi$ we denote that ϕ holds in the state g of the model M given the valuation υ. We omit the model symbol M, where this does not lead to ambiguity.

Definition 3 CTLPK semantics. Let M be a model and υ be a valuation of the group variables. The relation $\models_\upsilon$ is defined recursively as follows:

- $g \models_\upsilon p$ iff $p \in \mathcal{L}(g)$ for $p \in \mathcal{PV}$,
- $g \models_\upsilon \neg\phi$ iff $g \not\models_\upsilon \phi$,
- $g \models_\upsilon \phi \vee \psi$ iff $g \models_\upsilon \phi$ or $g \models_\upsilon \psi$,
- $g \models_\upsilon EX\phi$ iff there exists a path π starting at g, such that $\pi(1) \models_\upsilon \phi$,
- $g \models_\upsilon EG\phi$ iff there exists a path π starting at g, such that $\pi(i) \models_\upsilon \phi$ for all $i \geq 0$,
- $g \models_\upsilon E\psi U\phi$ iff there exists a path π starting at g, such that $\pi(j) \models_\upsilon \phi$ for some $j \geq 0$, and $\pi(i) \models_\upsilon \psi$ for all $0 \leq i < j$,
- $g \models_\upsilon K_i\phi$ iff for all $g' \in G$ if $g \sim_i g'$, then $g' \models_\upsilon \phi$,
- $g \models_\upsilon Z_\Gamma\phi$ iff for all $g' \in G$ if $g \sim_\Gamma^Z g'$, then $g' \models_\upsilon \phi$, where $Z \in \{E, D, C\}$,
- $g \models_\upsilon K_Y\phi$ iff $g \models_\upsilon K_i\phi$, where $\{i\} = \upsilon(Y)$,
- $g \models_\upsilon Z_Y\phi$ iff $g \models_\upsilon Z_{\upsilon(Y)}\phi$, where $Z \in \{E, D, C\}$.

We say that ϕ holds in a model M under valuation υ (denoted $M \models_\upsilon \phi$) if $M, g^0 \models_\upsilon \phi$.

Note that if ϕ is a ground formula (i.e., a CTLK formula), then $M, g \models_\upsilon \phi$ does not depend on the choice of υ.

3. GROUP SYNTHESIS FOR CTLPK

Given the semantics of CTLPK, a question that arises is, given a formula ϕ, how to determine all valuations υ for the group variables in ϕ such that all and only the ground instances of ϕ, obtained by substituting the group variables with concrete group values, are satisfied on the given model.

Formally, given a formula ϕ and a model M, we define a function $f_\phi : G \to 2^{GroupVals}$, returning at every global state the set of valuations for all the group variables to concrete group values such that $M, g \models_\upsilon \phi$ iff $\upsilon \in f_\phi(g)$.

In the rest of this section we present the algorithm $Synth_\phi$, a provably sound and complete procedure for constructing f_ϕ. The procedure is applied recursively bottom-up from the atoms to the out-most operators and is defined inductively. The correctness of the overall result follows from the correctness of the procedure for each individual modality.

In the following examples, we only consider formulae that contain a single group variable. We write $\{Y \rightsquigarrow \{A_1, \dots, A_m\}\}$ to represent the set of valuations $\{\upsilon_1, \dots, \upsilon_m\}$ s.t. $\upsilon_i(Y) = A_i$, for all $1 \le i \le m$.

Throughout this section we assume that the model M is fixed.

3.1 Boolean Operations and Non-parametric Modalities

We begin by defining the function f_ϕ for CTLPK modalities which do not contain group variables (i.e., the operators from CTLK).

Atomic Propositions.

For an atomic proposition $p \in \mathcal{PV}$, the function f_p is defined as follows:

$$f_p(g) = \begin{cases} GroupVals & \text{if } p \in \mathcal{L}(g), \\ \emptyset & \text{otherwise.} \end{cases}$$

For an atomic proposition, $M, g \models_\upsilon p$ does not depend upon υ; therefore for a state g such that $p \in \mathcal{L}(g)$, $f_p(g)$ is simply the set of all possible group valuations.

Example 1 (Atomic Propositions). Consider the simple three state, two-agent interpreted system model as illustrated in Figure 1. In this model we have $\mathcal{PV} = \{p, q\}$ where $\mathcal{L}(w_0) = \mathcal{L}(w_1) = \{p\}$ and $\mathcal{L}(w_2) = \{q\}$. Dashed lines labelled with an agent index represent the indistinguishability relation for that agent; solid lines labelled with 't' represent temporal transitions.

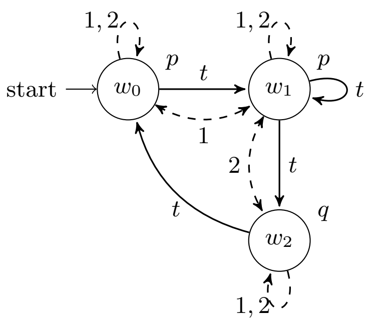


Figure 1: The interpreted system of Examples 1–5 and 7.

In this two-agent example with only one group variable we have that $GroupVals = \{Y \rightsquigarrow \{\{1\}, \{2\}, \{1,2\}\}\}$. It is straightforward to observe that:

$$f_p(g) = \begin{cases} \{Y \rightsquigarrow \{\{1\}, \{2\}, \{1,2\}\}\} & \text{for } g \in \{w_0, w_1\}, \\ \emptyset & \text{if } g = w_2. \end{cases}$$

Negation.

Given f_ϕ, to construct the function $f_{\neg\phi}$ we complement the groups defined in f_ϕ:

$$f_{\neg\phi}(g) = GroupVals \setminus f_\phi(g)$$

As per the definition of f_ϕ, we have that for each state g and valuation of group variables $\upsilon \in GroupVals$ we have $M, g \models_\upsilon \phi$ iff $\upsilon \in f_\phi(g)$. This is equivalent to the fact that for each state g and valuation $\upsilon \in GroupVals$ we have that $M, g \not\models_\upsilon \phi$ iff $\upsilon \notin f_\phi(g)$, which in turn is equivalent to $\upsilon \in GroupVals \setminus f_\phi(g)$, i.e., $\upsilon \in f_{\neg\phi}(g)$.

In presented algorithms we assume the existence of a subroutine, denoted *Complement*, realising the above operation, i.e., $Complement(f_\phi) = f_{\neg\phi}$.

Example 2 (Negation). Consider the model from Example 1. After computing the complement of f_p we obtain:

$$f_{\neg p}(g) = \begin{cases} \emptyset & \text{for } g \in \{w_0, w_1\}, \\ \{Y \rightsquigarrow \{\{1\}, \{2\}, \{1,2\}\}\} & \text{if } g = w_2. \end{cases}$$

Disjunction.

For each $\phi, \psi \in$ CTLPK, we have that for each $g \in G$, $f_{\phi\vee\psi}(g) = f_\phi(g) \cup f_\psi(g)$. It can easily be seen that $s \models_\upsilon \phi \vee \psi$ iff $s \models_\upsilon \phi$ or $s \models_\upsilon \psi$. This is equivalent to $\upsilon \in f_\phi(s)$ or $\upsilon \in f_\psi(s)$.

Temporal Operators.

Intuitively, a temporal transition does not alter the group assignments in f_ϕ that hold at a given successor. Therefore, for $EX\phi$ we take the union of all assignments for ϕ from each next state:

$$f_{EX\phi}(g) = \bigcup_{\{g' \in G \mid (g, g') \in T\}} f_\phi(g').$$

We now consider the case of $EG\phi$. Note that $f_{EG\phi}(g) = f_\phi(g) \cap f_{EXEG\phi}(g)$, for each $g \in G$. Therefore, in a similar way to the non-parametric case [2, 6], $f_{EG\phi}$ can be obtained through a fixed-point calculation.

Similarly, for $E\phi U\phi$ we have that $f_{E\phi U\psi}(g) = f_\psi(g) \cup (f_\phi(g) \cap f_{EXE\phi U\psi}(g))$. Again, such a calculation can be performed as a fixed-point.

Example 3 (Temporal Operators). Let us consider the formula $EXE_Y p$ that is to be evaluated at the initial state w_0 of model presented in Figure 1. Let us also assume that the function $f_{E_Y p}$ has been correctly calculated and is as follows (we illustrate its construction in Example 4):

- $f_{E_Y p}(w_0) = \{Y \rightsquigarrow \{\{1\}, \{2\}, \{1,2\}\}\}$
- $f_{E_Y p}(w_1) = \{Y \rightsquigarrow \{\{1\}\}\}$
- $f_{E_Y p}(w_2) = \emptyset$

From the construction of $EX\phi$ (where $\phi = E_Y p$) presented previously we have that:

- $f_{EXE_Y p}(w_0) = f_{E_Y p}(w_1) = \{Y \rightsquigarrow \{\{1\}\}\}$
- $f_{EXE_Y p}(w_1) = f_{E_Y p}(w_1) \cup f_{E_Y p}(w_2) = \{Y \rightsquigarrow \{\{1\}\}\}$
- $f_{EXE_Y p}(w_2) = f_{E_Y p}(w_0) = \{Y \rightsquigarrow \{\{1\}, \{2\}, \{1,2\}\}\}$

Finally, we evaluate $f_{EXE_Y p}$ at the state w_0 and obtain the single satisfiable valuation of $\upsilon = \{Y \rightsquigarrow \{\{1\}\}\}$. Such an assignment would correspond to the concrete formula $EXE_\Gamma p$, where $\Gamma = \{1\}$. This formula can easily be seen to hold at w_0.

Epistemic Operators.
The set of assignments that make $K_i\phi$ hold at g is equal to the intersection of all sets of assignments that make ϕ hold at any state i-distinguishable from g. Therefore, given the formula $K_i\phi \in$ CTLPK and a global state $g \in G$ we have:

$$f_{K_i\phi}(g) = \bigcap_{\{g' \in G \mid g \sim_i g'\}} f_\phi(g').$$

For the operators E_Γ, D_Γ and C_Γ we follow the same construction, but use the relations $\sim^E_\Gamma$, $\sim^D_\Gamma$, $\sim^C_\Gamma$ respectively.

3.2 Group Synthesis for Parametric Epistemic Operators

We now present a methodology for synthesising group assignments for the parametric epistemic modalities. As is common in techniques for calculating the satisfaction of epistemic formulae (see [12]), the following algorithms rely on the use of the dual for the associated modality being evaluated.

We begin by presenting the synthesis technique for the parametric "everybody knows" modality ($E_Y\phi$). As individual knowledge is a special case of everybody knows (i.e., where the set Γ for E_Γ consists of a single agent i), we delay this presentation until after $E_Y\phi$.

Synthesis for Everybody Knows.
To calculate the parametric assignments for $E_Y\phi$, we need to define the *existential group pre-image* between two global states g and g'. We denote this as $Link^{\exists}_Y(g,g')$ and it consists of all group valuations υ such that there exists an $i \in \upsilon(Y)$ and $g \sim_i g'$. Formally:

$$Link^{\exists}_Y(g,g') = \{\upsilon \in GroupVals \mid g \sim_i g' \text{ for some } i \in \upsilon(Y)\}.$$

It is important to note that $\sim_i$ is an equivalence relation, thus from its reflexivity we have that $Link^{\exists}_Y(g,g) = GroupVals$ and from its symmetry $Link^{\exists}_Y(g,g') = Link^{\exists}_Y(g',g)$ follows.

Algorithm 1 $Synth_E\,(f_\phi, Y)$

```
Input: $f_\phi \in \left(2^{GroupVals}\right)^G$
Output: $f_{E_Y\phi} \in \left(2^{GroupVals}\right)^G$
1: $f := Complement\,(f_\phi)$
2: $h := \emptyset$
3: for all $g \in G$ do
4:     $h(g) := \bigcup_{g' \in G} \left(Link^{\exists}_Y\,(g,g') \cap f\,(g')\right)$
5: end for
6: return $Complement\,(h)$
```

Lemma 4 *(Correctness of Synth$_E$). Let $M = (G, g^0, T, \sim_1, \ldots, \sim_n)$ be an interpreted system, $g \in G$, and $\phi \in$ CTLPK. Then, $M, g \models_\upsilon E_Y\phi$ iff $\upsilon \in Synth_E(f_\phi, Y)(g)$.*

Proof. Using the definition of $Link^{\exists}_Y$ and the inductive assumption on f_ϕ, we prove correctness of $Synth_E\,(f_\phi, Y)$. At the end of the loop between Lines 3 and 5 in Algorithm 1, for each global state $g \in G$, the set $h(g)$ consists of all the assignments υ such that there exists $i \in \upsilon(Y)$ and there exists $g' \in G$ where $g \sim_i g'$ and $M, g' \models_\upsilon \neg\phi$.

As such, $\upsilon \in h(g)$ iff $M, g \models_\upsilon \overline{E}_Y\neg\phi$, where $\overline{E}_Y\phi$ is defined as $\neg E_Y\neg\phi$. By taking the complement of h, we obtain the set of all assignments υ, which map each global state g to a set of valuations of group variables, such that:

$$\begin{aligned} & M, g \not\models_\upsilon \overline{E}_Y\neg\phi \text{ (def. of complement)} \\ \Leftrightarrow\ & M, g \models_\upsilon \neg\overline{E}_Y\neg\phi \text{ (def. of } \not\models) \\ \Leftrightarrow\ & M, g \models_\upsilon E_Y\phi \text{ (def. of } \overline{E}_Y). \end{aligned}$$

□

Example 4 (Everybody Knows). We now show that $f_{E_Y p}$ from Example 3 is correctly synthesised. At the first line of Algorithm 1, we take the complement of f_p (i.e., $f_{\neg p}$) and store it in the variable f. As such, the function held in the variable f contains the assignments $f(w_0) = f(w_1) = \emptyset$ and $f(w_2) = \{Y \rightsquigarrow \{\{1\}, \{2\}, \{1,2\}\}\}$ (see Example 2).

We now show the construction of $Link^{\exists}_Y(g,g')$ for all pairs of states. These are shown below (we omit the symmetric cases):

$$\begin{aligned} Link^{\exists}_Y(w_i,w_i) &= \{Y \rightsquigarrow \{\{1\}, \{2\}, \{1,2\}\}\}, \text{ for } i \in \{0,1,2\} \\ Link^{\exists}_Y(w_0,w_1) &= \{Y \rightsquigarrow \{\{1\}, \{1,2\}\}\} \\ Link^{\exists}_Y(w_1,w_2) &= \{Y \rightsquigarrow \{\{2\}, \{1,2\}\}\} \\ Link^{\exists}_Y(w_0,w_2) &= \emptyset \end{aligned}$$

At Line 4, we use $Link^{\exists}_Y(g,g')$ and $f(g')$ to construct $h(g)$. Therefore we need to compute $Link^{\exists}_Y(g,g') \cap f(g')$ for all pairs of states $g, g' \in \{w_0, w_1, w_2\}$. Notice that if $g' \in \{w_0, w_1\}$ then $f(g') = \emptyset$. Similarly, if $g = w_0$ and $g' = w_2$ then $Link^{\exists}_Y(g,g') = \emptyset$. This leaves only two remaining, non-empty cases:

$$\begin{aligned} Link^{\exists}_Y(w_1,w_2) \cap f(w_2) &= \{Y \rightsquigarrow \{\{2\}, \{1,2\}\}\} \\ Link^{\exists}_Y(w_2,w_2) \cap f(w_2) &= \{Y \rightsquigarrow \{\{1\}, \{2\}, \{1,2\}\}\} \end{aligned}$$

Next, at Line 4, we calculate the function h by assigning to each $h(g)$ the union of $Link^{\exists}_Y(g,g') \cap f(g')$ for all $g' \in \{w_0, w_1, w_2\}$. The function h is therefore as follows:

- $h(w_0) = \emptyset$
- $h(w_1) = \{Y \rightsquigarrow \{\{2\}, \{1,2\}\}\}$
- $h(w_2) = \{Y \rightsquigarrow \{\{1\}, \{2\}, \{1,2\}\}\}$

Finally, we take the complement of h to calculate the function $f_{E_Y p}$. The result can be seen below:

- $f_{E_Y p}(w_0) = \{Y \rightsquigarrow \{\{1\}, \{2\}, \{1,2\}\}\}$
- $f_{E_Y p}(w_1) = \{Y \rightsquigarrow \{\{1\}\}\}$
- $f_{E_Y p}(w_2) = \emptyset$

These assignments mean that $E_Y p$ can be satisfied at the state w_0 with any assignment to the variable Y and at the state w_1 with the assignment $Y = \{1\}$. However, there is no substitution that makes $E_Y p$ hold at w_2.

Synthesis of Individual Knowledge.

As previously stated, individual knowledge is a special case of everybody knows, where each group assignment consists of only a single agent. We adapt the set $Link^{\exists}_{Y}$ as below:

$$Link^{\text{indv.}}_{Y}(g, g') = \{v \in GroupVals \mid v(Y) = \{i\} \text{ and } g \sim_i g'\}.$$

The set $Link^{\text{indv.}}_{Y}$ contains only those group valuations that assign to Y a single agent. To compute $f_{K_Y \phi}$, we use Algorithm $Synth_K$; this algorithm can be simply constructed by substituting $Link^{\exists}_{Y}$ with $Link^{\text{indv.}}_{Y}$ on Line 4 of Algorithm 1.

Corollary 5 ***(Correctness of*** $Synth_K$***).*** *Let* $M = (G, g^0, T, \sim_1, \dots, \sim_n)$ *be an interpreted system,* $g \in G$, *and* $\phi \in CTLPK$. *Then,* $M, g \models_v K_Y \phi$ *iff* $v \in Synth_K(f_\phi, Y)(g)$.

The proof is straightforward and can easily be obtained by replacing $\overline{E}_Y$ with $\overline{K}_Y$ in the proof of Lemma 4.

Example 5 (Individual Knowledge). We demonstrate how to construct $f_{K_Y p}$ for the model in Example 3. The only difference between $Synth_K$ and Algorithm 1 is that we substitute $Link^{\exists}_{Y}(g, g')$ for $Link^{\text{indv.}}_{Y}(g, g')$. The latter consists of all group valuations $v \in Link^{\exists}_{Y}(g, g')$ such that $v(Y)$ is a single-element group.

Using the values for $Link^{\exists}_{Y}(g, g')$ from Example 4, and selecting only those group assignments consisting of a single agent, we obtain:

$$Link^{\text{indv.}}_{Y}(w_1, w_2) \cap f(w_2) = \{Y \rightsquigarrow \{\{2\}\}\}$$
$$Link^{\text{indv.}}_{Y}(w_2, w_2) \cap f(w_2) = \{Y \rightsquigarrow \{\{1\}, \{2\}\}\}$$

where $Link^{\text{indv.}}_{Y}(g, g') = \emptyset$ for the remaining cases.

Taking the union of $Link^{\text{indv.}}_{Y}(g, g') \cap f(g')$ for all $g, g' \in G$, and then complementing, we obtain the function $f_{K_Y p}$ as:

- $f_{K_Y p}(w_0) = \{Y \rightsquigarrow \{\{1\}, \{2\}\}\}$
- $f_{K_Y p}(w_1) = \{Y \rightsquigarrow \{\{1\}\}\}$
- $f_{K_Y p}(w_2) = \emptyset$

It can easily be seen that for all states $g \in G$ the set $f_{K_Y p}(g)$ consists of exactly those valuations from $f_{E_Y p}(g)$ that assign to Y groups consisting of one agent only.

Synthesis of Distributed Knowledge.

Given two global states $g, g' \in G$ and a group $Y \in Groups$, we define:

$$Link^{\forall}_{Y}(g, g') = \{v \in GroupVals \mid g \sim_i g' \text{ for all } i \in v(Y)\}.$$

In contrast to $Link^{\exists}_{Y}(g, g')$, the set $Link^{\forall}_{Y}(g, g')$ consists of all assignments v, such that for all agents $i \in v(Y)$, i considers g and g' as indistinguishable (i.e., the group Y considers g and g' indistinguishable).

Algorithm 2 $Synth_D\ (f_\phi, Y)$

```
Input: $f_\phi \in \left(2^{GroupVals}\right)^G$
Output: $f_{D_Y \phi} \in \left(2^{GroupVals}\right)^G$
1: $f := Complement\ (f_\phi)$
2: $h := \emptyset$
3: for all $g \in G$ do
4:     $h(g) := \bigcup_{g' \in G} \left(Link^{\forall}_{Y}\ (g, g') \cap f\ (g')\right)$
5: end for
6: return  $Complement\ (h)$
```

Lemma 6 ***(Correctness of*** $Synth_D$***).*** *Let* $M = (G, g^0, T, \sim_1, \dots, \sim_n)$ *be an interpreted system,* $g \in G$, *and* $\phi \in CTLPK$. *Then,* $M, g \models_v D_Y \phi$ *iff* $v \in Synth_D(f_\phi, Y)(g)$.

PROOF. We define the modality $\overline{D}_Y \phi$ as $\neg D_Y \neg \phi$; a global state g satisfies $\overline{D}_Y \phi$ under group valuation v if there exists a state g' such that $g \sim_D^{v(Y)} g'$ and $M, g \models_v \phi$.

At the end of Line 5 of Algorithm 2 we have that, for each global state $g \in G$, the set $h(g)$ consists of all assignments v such that there exists a global state $g' \in G$, where for all $i \in v(Y)$, $g \sim_i g'$ and $M, g' \models_v \neg\phi$. So we have $M, g \models_v \overline{D}_Y \neg\phi$.

By taking the complement of h at the end of Algorithm 2, we obtain the set of all assignments v, which map each global state g to a set of valuations of group variables, such that:

$$\begin{aligned} & M, g \not\models_v \overline{D}_Y \neg\phi \text{ (def. of complement)} \\ \Leftrightarrow\ & M, g \models_v \neg\overline{D}_Y \neg\phi \text{ (def. of } \not\models) \\ \Leftrightarrow\ & M, g \models_v D_Y \phi \text{ (def. of } \overline{D}_Y). \end{aligned}$$

Therefore, we have that for all $g \in G$, for all valuations $v \in f_{D_Y \phi}(g)$ iff $M, g \models_v D_Y \phi$.

□

Example 6 (Distributed Knowledge). We now adapt the interpreted system from Figure 1 by making all the states indistinguishable for Agent 2 (see Figure 2). As before, we have $\mathcal{PV} = \{p, q\}$ and the states are labelled such that $\mathcal{L}(w_0) = \mathcal{L}(w_1) = \{p\}$ and $\mathcal{L}(w_2) = \{q\}$.

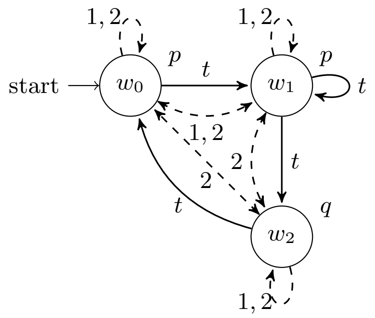


Figure 2: The interpreted system of Example 6.

The introduction of new epistemic links in the model does not change the values of either f_p or $f_{\neg p}$. Therefore, the variable f (i.e., $f_{\neg p}$) at Line 1 of Algorithm 2 is such that $f(w_0) = f(w_1) = \emptyset$ and $f(w_2) = \{Y \rightsquigarrow \{\{1\}, \{2\}, \{1,2\}\}\}$.

Recall that $Link^{\forall}_{Y}(g, g')$ consists of all group valuations assigning to variable Y groups consisting solely of agents that cannot distinguish between g and g'. We now build $Link^{\forall}_{Y}(g, g')$ for all the pairs of states. Again we omit the symmetric cases:

$$Link^{\forall}_{Y}(w_i, w_i) = \{Y \rightsquigarrow \{\{1\}, \{2\}, \{1,2\}\}\} \text{ for all } i \in \{0,1,2\}$$
$$Link^{\forall}_{Y}(w_0, w_1) = \{Y \rightsquigarrow \{\{1\}, \{2\}, \{1,2\}\}\}$$
$$Link^{\forall}_{Y}(w_1, w_2) = Link^{\forall}_{Y}(w_0, w_2) = \{Y \rightsquigarrow \{\{2\}\}\}$$

In Line 4 of Algorithm 2 for each state g we compute $h(g)$ as the union of sets $Link^{\forall}_{Y}(g, g') \cap f(g')$ over all $g' \in \{w_0, w_1, w_2\}$. Again, notice that if $g' \in \{w_0, w_1\}$ then $Link^{\forall}_{Y}(g, g') \cap f(g') = \emptyset$, thus we need to consider only the following cases:

$$Link^{\forall}_{Y}(w_0, w_2) \cap f(w_2) = \{Y \rightsquigarrow \{\{2\}\}\}$$
$$Link^{\forall}_{Y}(w_1, w_2) \cap f(w_2) = \{Y \rightsquigarrow \{\{2\}\}\}$$
$$Link^{\forall}_{Y}(w_2, w_2) \cap f(w_2) = \{Y \rightsquigarrow \{\{1\}, \{2\}, \{1,2\}\}\}$$

The function evaluated in the loop between Lines 3–5 and held in variable h is equal to $f_{\overline{D}_Y \neg p}$; thus after complementing in the return statement we obtain:

- $f_{D_Y p}(w_0) = \{Y \rightsquigarrow \{\{1\}, \{1,2\}\}\}$
- $f_{D_Y p}(w_1) = \{Y \rightsquigarrow \{\{1\}, \{1,2\}\}\}$
- $f_{D_Y p}(w_2) = \emptyset$

Synthesis of Common Knowledge.

Recall from [4] that $C_\Gamma \phi \Leftrightarrow E_\Gamma(\phi \wedge C_\Gamma \phi)$, for any $\Gamma \subseteq$ *Agents*. We can use this equivalence to compute the set of states satisfying $C_\Gamma \phi$ by reasoning through existential pre-images of the epistemic relations for E_Γ. We use the following algorithm, an extension of the non-parametric version presented in [12].

The synthesis of parametric common knowledge employs a similar observation, i.e., that C_Y is the fixed-point of E_Y.

Algorithm 3 *Synth*$_C$ (f_ϕ, Y)

```
Input: $f_\phi \in \left(2^{GroupVals}\right)^G$
Output: $f_{C_Y \phi} \in \left(2^{GroupVals}\right)^G$
1: $f := \emptyset$
2: $h := Complement\,(f_\phi)$
3: while $f \neq h$ do
4:     $f := h$
5:     for all $g \in G$ do
6:         $h(g) := \bigcup_{g' \in G} \left(Link^{\exists}_Y\,(g, g') \cap f\,(g')\right)$
7:     end for
8: end while
9: return  $Complement\,(h)$
```

Lemma 7 ***(Correctness of Synth$_C$).***
Let $M = (G, g^0, T, \sim_1, \ldots, \sim_n)$ be an interpreted system, $g \in G$, and $\phi \in CTLPK$. Then, $M, g \models_\upsilon C_Y \phi$ iff $\upsilon \in Synth_C(f_\phi, Y)(g)$.

PROOF. As usual, let $\overline{E}^0_Y \phi = \phi$, $\overline{E}^{i+1}_Y \phi = \overline{E}_Y(\overline{E}^i_Y \phi)$ for all $i \geq 0$ and $\overline{C}_Y \phi = \neg C_Y \neg \phi$. Since $C_Y \phi \Leftrightarrow E_Y\,(\phi \wedge C_Y \phi)$, we have that for each state g and each group valuation $\upsilon \in$ *GroupVals*:

$$M, g \models_\upsilon \overline{C}_Y \phi \text{ iff } M, g \models_\upsilon \bigvee_{i=0}^{j} \overline{E}^i_Y \phi \text{ for some } j \geq 0. \qquad (\star)$$

Observe now that in Algorithm 3, prior to the execution of the loop between Lines 3–8, the function h is equivalent to $f_{\neg\phi}$. Given Lemma 4 (i.e., the correctness of *Synth*$_E$), after the first iteration of this inner loop h evaluates to $f_{\neg\phi \vee \overline{E}_Y \neg\phi}$. After the j-th iteration of the main body (Lines 3–8), the function h evaluates to

$$h = f_{\bigvee_{i=0}^{j} \overline{E}^i_Y \neg\phi}.$$

Given that we work on finite models, h eventually reaches a fixed-point. At that point $h = f_{\bigvee_{j=0}^{k} \overline{E}^j_Y \neg\phi}$, where k is the smallest value of j in $\star$. Therefore we have that $h = f_{\overline{C} \neg\phi}$. After taking the complement we have:

$$\begin{aligned} & M, g \not\models_\upsilon \overline{C}_Y \neg\phi \text{ (def. of complement)} \\ \Leftrightarrow\ & M, g \models_\upsilon \neg\overline{C}_Y \neg\phi \text{ (def. of } \not\models) \\ \Leftrightarrow\ & M, g \models_\upsilon C_Y \phi \text{ (def. of } \overline{C}_Y). \end{aligned}$$

This concludes the proof for $C_Y \phi$.

□

Example 7 (Common Knowledge). We now present how to synthesise $f_{C_Y p}$ for the model presented in Figure 1. As before, let $\mathcal{PV} = \{p, q\}$ and the states be labelled such that $\mathcal{L}(w_0) = \mathcal{L}(w_1) = \{p\}$ and $\mathcal{L}(w_2) = \{q\}$.

Note that in the first run of the 3–8 loop of Algorithm 3 the result of the evaluation of the function held in h variable is equal to the result of the evaluation in 3–5 loop of Algorithm 1. We can therefore reuse the values for h and $Link^{\exists}_Y(g, g')$ from Example 4 to compute:

$$\begin{aligned} & Link^{\exists}_Y(w_0, w_1) \cap f(w_1) = \{Y \rightsquigarrow \{\{1,2\}\}\} \\ & Link^{\exists}_Y(w_0, w_2) \cap f(w_2) = \emptyset \\ & Link^{\exists}_Y(w_1, w_1) \cap f(w_1) = Link^{\exists}_Y(w_1, w_2) \cap f(w_2) \\ & = Link^{\exists}_Y(w_2, w_1) \cap f(w_1) = \{Y \rightsquigarrow \{\{2\}, \{1,2\}\}\} \\ & Link^{\exists}_Y(w_2, w_2) \cap f(w_2) = \{Y \rightsquigarrow \{\{1\}, \{2\}, \{1,2\}\}\} \end{aligned}$$

Note that f is equal to h due to the substitution in Line 4 and the remaining cases are equal to $Link^{\exists}_Y(g, w_0) \cap f(w_0)$, which yields the empty set.

In the second run of the while loop we again calculate $h(g)$ for each state g by computing the union of all $Link^{\exists}_Y(g, g') \cap f(g')$ for each $g' \in \{w_0, w_1, w_2\}$ (Line 6). The result is as follows:

- $h(w_0) = \{Y \rightsquigarrow \{\{1,2\}\}\}$
- $h(w_1) = \{Y \rightsquigarrow \{\{2\}, \{1,2\}\}\}$
- $h(w_2) = \{Y \rightsquigarrow \{\{1\}, \{2\}, \{1,2\}\}\}$

It can be easily seen that the next run of the loop does not change the value of h, so the fixed-point is reached and we exit the while loop. The function held in variable h is equal to $f_{\overline{C}_Y \neg p}$, therefore after the complement in the return statement we obtain:

- $f_{C_Y p}(w_0) = \{Y \rightsquigarrow \{\{1\}, \{2\}\}\}$
- $f_{C_Y p}(w_1) = \{Y \rightsquigarrow \{\{1\}\}\}$
- $f_{C_Y p}(w_2) = \emptyset$

which concludes our example.

The following algorithm is presented to provide the entry point for calculating the function f_ϕ for given $\phi \in$ CTLPK by recursively calling previously introduced subroutines.

Algorithm 4 *Synth*$_{\text{CTLPK}}$ (ϕ)

```
Input: $\phi \in$ CTLPK
Output: $f_\phi \in \left(2^{GroupVals}\right)^G$
1: if $\phi = Z_Y \psi$ then
2:     return  $Synth_Z(Synth_{\text{CTLPK}}(\psi), Y)$
3: else /* non-parametric mod. omitted for simplicity */
4:     return  $f_\phi$
5: end if
```

Theorem 8 (Group Synthesis for CTLPK).
For each model M, for all global states $g \in G$ and for all formulae ϕ in CTLPK, we have that $M, g \models_\upsilon \phi$ iff $\upsilon \in f_\phi(g)$.

PROOF. The validity of the theorem follows immediately from the previous treatment of propositions, Boolean and temporal operators, Lemmas 4, 6 and 7 and Corollary 5. □

4. EVALUATION

We have implemented the presented parametric approach as an experimental extension to the open-source model checker MCMAS [9]. A GNU GPL licenced release is available from `http://vas.doc.ic.ac.uk/tools/mcmas_parametric/`. As MCMAS is a symbolic model checker, the satisfiable group valuations are also stored symbolically. We compare this to a naïve approach that iteratively checks each possible group assignment for satisfiability. Observe that for n agents and m group variables, there are $(2^n - 1)^m$ unique group assignments.

We carry out this comparison using two benchmarks.

4.1 Dining Cryptographers

We first consider Chaum's Dining Cryptographers protocol [1]; this problem has been widely studied with respect to multi-agent systems and temporal-epistemic logic [10].

We consider the following two parametric specifications (we write $paid_i$ to represent the proposition "diner i paid"):

- $\varphi_{DC1} = AG\,(paid_1 \rightarrow (C_Y\,(paid_1)))$
- $\varphi_{DC2} = AG\,(paid_1 \rightarrow (C_Y\,(paid_1 \wedge \neg D_Z\,(\neg paid_2))))$

The first specification φ_{DC1} expresses "if diner one paid, then the group Y has common knowledge that diner one paid". This specification is satisfied only for the valuation $Y = \{Diner\_1\}$. The formula φ_{DC2} extends the first formula by additionally stating that "the group Y also has common knowledge that the group Z does not have distributed knowledge that diner two did not pay". Considering only the case in which diner one paid, this specification is satisfied when $Z = GroupVals \setminus \{Diner\_1, Diner\_2\}$ (Y is as for φ_{DC1}). The group Z cannot include $Diner\_2$, because if diner one paid, then diner two knows (individually) that he did not.

The experimental results for parameter synthesis for these formulae over models of varying size can be seen in Table 1. The values were collected over three runs, with a timeout of one hour per run. The machine employed for these benchmarks was an Intel Core 2 Duo processor 3.00 GHz, with a 6144 KiB cache, and ran 32-bit Fedora 14, kernel 2.6.35.14. These results show that the parametric approach can, memory permitting, perform synthesis faster than the naïve approach. This is exemplified for 18 diners and the formula φ_{DC2}, where parametric verification completed in under 11 minutes but naïve did not finish within the hour.

For a model containing 14 diners, the construction of the reachable state space, regardless of implementation or formula, exhibited an unusually high run-time. This is reflected in the TIMEOUT and MEMOUT results for this sized model. We believe that this was caused by BDD reordering within the CUDD library utilised by MCMAS.

4.2 IEEE Token Ring Network

We now compare the parametric and naïve approaches using the industry standard IEEE token ring bus network. In the comparison that follows, we automated the injection of faults into the model, following the approach of [3]; this allows for the automatic analysis of fault-diagnosability properties. We briefly summarise the scenario below; for a complete description we refer the reader to [3].

The IEEE token ring protocol connects n nodes in a ring topology; data moves between nodes on the network in a clockwise fashion. Access is granted to nodes on the network in the form of a token; this is passed from node to node. Tokens are issued onto the network from an "active monitor". To detect faults, tokens contain a "time to live" field, initialised to the maximum time that a token would take to circulate the whole network and counting down to zero. Should a token fail to circulate back to the active monitor within the given time-frame, it is deduced that a fault has occurred on the network.

We consider instantiations of the network where the first node wishes to transmit a data token to the final node. Consequently, data needs to pass through every single intermediate node on the system.

Using a modified version of the fault injector from [3], we inserted two types of non-deterministic faults: even nodes stop sending tokens and odd nodes stop receiving tokens. The properties verified are shown in Table 2. To exemplify: φ_{TR2} states that "there exists a future state where the group Y has common knowledge that the group Z does not possess common knowledge that faults have not been injected".

Table 3 shows the comparison between the parametric and naïve approach. These results demonstrate the benefits of parametric verification.

5. CONCLUSIONS

We have introduced a novel parametric temporal-epistemic logic, as well as presenting a sound and complete approach for parameter synthesis. Using two non-trivial examples, we have shown that a symbolic implementation of the parametric technique can be more efficient than a brute force approach.

The results corroborate the expected: the parametric technique sacrifices memory efficiency for tractability. The current experimental results seem favourable to the parametric approach. However, the results also demonstrate that there exist models where the naïve technique can complete synthesis, while the parametric approach runs out of memory.

We are interested in applying the parametric technique to a wider range of industrial scenarios, for example, those where the synthesised groups can be used to define cliques of agents during the design and implementation phase.

Acknowledgements.

The authors acknowledge support from: the EPSRC (grants EP/E02727X/1 and EP/I00520X/1); the Polish National Science Centre (grant No. 2011/01/B/ST6/01477); and the SRI PAS sponsored International Ph.D. Projects in Intelligent Computing.

6. REFERENCES

[1] D. Chaum. The Dining Cryptographers Problem. *J. Cryptol.*, 1:65–75, 1988.

[2] E. M. Clarke, Jr., O. Grumberg, and D. A. Peled. *Model Checking.* MIT Press, Cambridge, MA, USA, 1999.

[3] J. Ezekiel and A. Lomuscio. A Methodology for Automatic Diagnosability Analysis. In *Proc. of ICFEM'10*, pages 549–564, 2010.

[4] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge.* MIT Press, 1995.

Table 1: Comparison for the Dining Cryptographers

Model		Formula	Group Valuations		Time (s)		Memory (KiB)	
Diners	States		Possible	# SAT	PARAMETRIC	NAÏVE	PARAMETRIC	NAÏVE
6	1,344	φ_{DC1}	63	1	0.084	0.064	11,500	10,504
		φ_{DC2}	3,969	15	0.155	0.777	13,184	10,504
10	33,792	φ_{DC1}	1,023	1	1.054	3.380	30,372	46,164
		φ_{DC2}	1,046,529	255	1.959	1,286.33	53,296	45,800
14	737,280	φ_{DC1}	16,383	1	TIMEOUT	317.093	TIMEOUT	138,951
		φ_{DC2} *	268,402,689	–*	MEMOUT	TIMEOUT	MEMOUT	TIMEOUT
18	$1.5 \cdot 10^7$	φ_{DC1}	262,143	1	324.034	731.034	$2.096 \cdot 10^6$	$1.747 \cdot 10^6$
		φ_{DC2}	68,718,952,449	65535	651.206	TIMEOUT	$2.606 \cdot 10^6$	TIMEOUT

* MCMAS-PARAMETRIC used over 3 GiB of RAM and MCMAS-NAÏVE failed to finish within the one-hour timeout.

Table 2: Diagnosability Properties for the Token Ring Protocol

Formula	Specification
φ_{TR1}	$EF\ C_Y \neg (\bigvee_{fault \in Faults} fault\_injected)$
φ_{TR2}	$EF\ C_Y \neg C_Z \neg (\bigvee_{fault \in Faults} fault\_injected)$
φ_{TR3}	$E[(C_Y \neg(\bigvee_{fault \in Faults} fault\_injected))\ U$ $(EF\ E_Z\ D_V \neg(\bigvee_{fault \in Faults} (fault\_injected \vee fault\_stopped)))]$

Table 3: Comparison for the Token Ring Network

Model		Formula	Group Valuations		Time (s)		Memory (KiB)	
Nodes	States		Possible	# SAT	PARAMETRIC	NAÏVE	PARAMETRIC	NAÏVE
4	1,116	φ_{TR1}	15	5	0.594	0.544	14,016	12,072
		φ_{TR2}	225	171	0.752	0.761	25,388	12,300
		φ_{TR3}	3,375	3,255	0.776	3.117	15,888	12,160
6	21,578	φ_{TR1}	63	7	1.434	1.209	23,684	23,680
		φ_{TR2}	3,969	3,571	2.321	7.285	55,588	23,680
		φ_{TR3}	250,047	232,407	11.368	408.478	40,752	23,680
8	336,632	φ_{TR1}	255	9	3.369	3.107	58,708	39,960
		φ_{TR2}	65,025	62,803	7.497	193.948	63,496	48,200
		φ_{TR3}	16,581,375	15,253.335	1,127.170	TIMEOUT†	62,316	TIMEOUT†
10	$7.769 \cdot 10^6$	φ_{TR1}	1,023	11	6.236	17.052	58,444	53,756
		φ_{TR2}	1,046,529	1,035,387	65.875	TIMEOUT‡	61,788	TIMEOUT‡
		φ_{TR3} *	1,070,599,167	–*	TIMEOUT*			
12	$1.157 \cdot 10^8$	φ_{TR1}	4,095	13	15.556	204.853	66,224	109,280$^{\bullet}$
		φ_{TR2}	16,769,025	16,715,947	1,423.12	TIMEOUT*	71,160	TIMEOUT*
		φ_{TR3} *	68,669,157,375	–*	TIMEOUT*			

† Calculated 8.35% of all the satisfiable groups within the timeout period.
‡ Calculated 73.92% of all the satisfiable groups within the timeout period.
$^{\bullet}$ Represents an anomalous result with MCMAS-NAÏVE/CUDD. The memory usage of MCMAS-NAÏVE should, under a correctly functioning system, always be lower than that of MCMAS-PARAMETRIC.
* Calculated 1.82% of all the satisfiable groups within the timeout period.
* Both MCMAS-PARAMETRIC and MCMAS-NAÏVE failed to halt-and-succeed within the one-hour time limit.

[5] P. Gammie and R. van der Meyden. MCK: Model Checking the Logic of Knowledge. In *Proc. of CAV'04*, pages 479–483, 2004.

[6] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning About Systems (Second Edition)*. Cambridge University Press, 2004.

[7] M. Kacprzak, W. Nabialek, A. Niewiadomski, W. Penczek, A. Pólrola, M. Szreter, B. Wozna, and A. Zbrzezny. VerICS 2007 - a Model Checker for Knowledge and Real-Time. *Fundam. Inform.*, 85(1-4):313–328, 2008.

[8] M. Knapik, W. Penczek, M. Szreter, and A. Pólrola. Bounded Parametric Verification for Distributed Time Petri Nets with Discrete-Time Semantics. *Fundam. Inform.*, 101(1–2):9–27, 2010.

[9] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A Model Checker for the Verification of Multi-Agent Systems. In *Proc. of CAV'09*, pages 682–688, 2009.

[10] R. v. d. Meyden and K. Su. Symbolic Model Checking the Knowledge of the Dining Cryptographers. In *Proc. of CSFW'04*, pages 280–291, 2004.

[11] W. Penczek and A. Lomuscio. Verifying Epistemic Properties of Multi-Agent Systems via Bounded Model Checking. *Fundam. Inform.*, 55(2):167–185, 2003.

[12] F. Raimondi and A. Lomuscio. Automatic Verification of Multi-Agent Systems by Model Checking via Ordered Binary Decision Diagrams. *J. Applied Logic*, 5(2):235–251, 2007.

[13] F. Wang. Timing Behavior Analysis for Real-Time Systems. In *Proc. of LICS'95*, pages 112–122, 1995.

A Logic of Revelation and Concealment

Wiebe van der Hoek Petar Iliev Michael Wooldridge
Department of Computer Science
University of Liverpool, UK
{wiebe, pvi, mjw}@liverpool.ac.uk

ABSTRACT

The last decade has been witness to a rapid growth of interest in logics intended to support reasoning about the interactions between knowledge and action. Typically, logics combining dynamic and epistemic components contain ontic actions (which change the state of the world, e.g., switching a light on) or epistemic actions (which affect the information possessed by agents, e.g., making an announcement). We introduce a new logic for reasoning about the interaction between knowledge and action, in which each agent in a system is assumed to perceive some subset of the overall set of Boolean variables in the system; these variables give rise to epistemic indistinguishability relations, in that two states are considered indistinguishable to an agent if all the variables visible to that agent have the same value in both states. In the dynamic component of the logic, we introduce actions $r(p, i)$ and $c(p, i)$: the effect of $r(p, i)$ is to reveal variable p to agent i; the effect of $c(p, i)$ is to conceal p from i. By using these dynamic operators, we can represent and reason about how the knowledge of agents changes when parts of their environment are concealed from them, or by revealing parts of their environment to them. Our main technical result is a sound and complete axiomatisation for our logic.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Modal Logic*; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Theory

Keywords

Modal logic, epistemic logic, dynamic epistemic logic, interpreted systems, knowledge and change

1. INTRODUCTION

Over the past decade, there has been a rapid growth of interest in logics intended for reasoning about the interaction between knowledge and action (see, e.g., [3] for extensive references). Such *Dynamic Epistemic Logics* make it possible to investigate many different dimensions along which action can interact with knowledge.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

For example, one recent and very active area of research at the interface of knowledge and action is the study of how communicative utterances such as public announcements change the knowledge of those that witness the utterance [3, 9].

Our aim in the present paper is to introduce a logic that is intended for representing an aspect of the relationship between knowledge and action that has not hitherto been considered: *how the knowledge of agents is changed by concealing parts of their environment from them, and revealing parts of their environment to them.* In more detail, we are concerned with scenarios in which we have a set of agents, where each agent i is associated with a set of Boolean variables V_i, the idea being that agent i can completely and correctly perceive the value of the variables in V_i. We refer to V_i as the *visibility set* of agent i. To represent what an agent knows in such a scenario, our logic uses conventional S5 epistemic modalities K_i, where $K_i\varphi$ means that agent i knows φ [4]. The semantics of epistemic modalities is defined via possible worlds, with the interpretation that two states s and s' are indistinguishable to agent i if the variables V_i have the same value in s and s'. To model revelation and concealment, our logic has dynamic modalites [7], in which we have atomic actions of the form $r(p, i)$ and $c(p, i)$, meaning *reveal p to i* and *conceal p from i*, respectively. The effect of performing the action $r(p, i)$ is that the variable p is added to i's visibility set; and the effect of performing the action $c(p, i)$ is that the variable p is removed from i's visibility set. For example, the following formula of our logic asserts that i doesn't know $p \wedge q$, but after revealing p to i, it does:

$$\neg K_i(p \wedge q) \wedge [r(p, i)]K_i(p \wedge q).$$

The remainder of the paper is structured as follows. We introduce the logic in the following section, beginning with an informal overview of the language, and some example formulae and their intended meaning. We then go on to present the formal syntax and semantics of the logic, and consider a detailed example, showing how the logic can be used to axiomatise properties of a multi-agent voting scenario. We then present the main technical result of the paper: an axiomatisation of the logic, for which we prove completeness using a type of canonical model construction. We conclude with some comments and issue for future work. Throughout the paper we assume some familiarity with modal, dynamic, and epistemic logics (see, e.g., [4, 6, 7, 2]).

2. THE LOGIC

We will begin with an informal overview of the logic, before presenting the formal syntax and semantics.

2.1 Overview

The *Logic of Revelation and Concealment* (LRC) is a combination of the well-known multi-agent epistemic logic S5$_n$ [4] with a specialised dynamic logic component [7], which is intended to allow us to reason about the effects of revealing variables to agents, and concealing variables from them. The language of LRC is parameterised by the following basic sets:

- A set $\mathcal{N} = \{1, \ldots, n\}$ of *agents*;
- A set $\Phi = \{p, q, \ldots\}$ of *Boolean variables*;
- A finite set of $Vis = \{v_1, \ldots, v_m\}$ of *visibility variables*;
- A set $\mathcal{A} = \{\alpha, \alpha', \ldots\}$ of *state changing actions*;
- A set $RC = \{r(v,i), c(v,i)\}$, where $i \in \mathcal{N}$, $v \in Vis$, of *revelation and concealment actions*; and
- a set of variables $SEES = \{sees_i(v) \mid i \in \mathcal{N}, v \in Vis\}$.

The language contains the usual Boolean operators of classical logic ($\wedge$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$). When we refer to propositional atoms or variables we mean $\Phi \cup Vis \cup SEES$. Formulae over those variables using only Boolean operators are called *objective*. To represent the knowledge possessed by agents in the system, we use indexed unary modalities K_i, where $i \in \mathcal{N}$, so that $K_i\varphi$ is intended to mean "agent i knows φ". The semantics of epistemic modalities is based on the interpreted systems model of knowledge [4]. Each agent $i \in \mathcal{N}$ is associated with a subset V_i of the variables Vis, with the idea being that the agent i is able to completely and correctly perceive the values of the variables V_i, but is not able to perceive the values of any other variables. We call V_i the *visibility set* of agent i. Thus, in the terminology of interpreted systems, V_i represents the *local state* of i. So, more precisely:

> $K_i\varphi$ means that, given the background knowledge of the agent and the variables V_i that he currently sees, the agent i can infer φ.

Note that the fact that $v \in V_i$ does not imply that i "controls" or has "write access" to i (cf. [11, 5, 10]): it simply means that i is able to see the value of v. Thus it could be that two different agents are able to see some of the same variables (i.e., we might have $v \in V_i$ and $v \in V_j$ for $i \neq j$). We require Vis to be finite for two reasons. The first is technical and it is reflected in the use of a particular axiom, (Ax37), in our axiomatic system, described later in the paper. This axiom helps us to enforce a certain property of the canonical model for our logic. The second reason is purely philosophical. If the agents we are interested in modelling have bounded observational and reasoning capabilities, then they can surely observe only finitely many features of their environment.

Within the object language, we can refer to the variables that an agent sees by using the primitive operators $sees_i(v)$, with the obvious meaning.

To represent actions and the effect that actions have on the system, we use a dynamic logic component, with program modalities $[\pi]$ ("after all executions of program $\pi\ldots$") and $\langle\pi\rangle$ ("after some execution of program $\pi\ldots$"). Programs π within dynamic modalities are constructed from *atomic actions*. Atomic actions in our language are of two types. First, we have a set $\mathcal{A}$ of *state changing actions*, typically denoted $\alpha, \alpha', \ldots$, which are essentially the same as atomic actions in conventional propositional dynamic logic [7]. The effect of performing such an action is to change the state of the system; we allow for the possibility that state changing actions have multiple possible outcomes.

In addition to the conventional PDL-style state-changing actions $\mathcal{A}$, in LRC we have a set RC of two additional types of atomic actions, $r(v,i)$ and $c(v,i)$, where $i \in \mathcal{N}$ and $v \in Vis$. The action $r(v,i)$ is read "reveal v to i", while $c(v,i)$ is read "conceal v from i". The effect of performing $r(v,i)$ is to *add the variable v to agent i's visibility set*, while the effect of $c(v,i)$ is to *remove v from i's visibility set*. These two atomic programs thus directly manipulate an agent's local state, and since what an agent knows is determined solely by its local state, they can also change what an agent knows. Notice, however, that actions $r(v,i)$ and $c(v,i)$ do *not* change the actual state of the system. In this sense, state changing actions $\mathcal{A}$ and visibility actions $r(v,i)$, $c(v,i)$ can be understood as causing changes to an agent's knowledge along two different dimensions: visibility actions $r(v,i)$ and $c(v,i)$ change an agent's visibility set but do not change the state of the system, while actions $\mathcal{A}$ change the state of the system but do not change an agent's visibility set.

Atomic actions are combined into complex programs using the usual program constructs of dynamic logic [7]: $\pi_1;\pi_2$ means "execute program π_1 and then execute program π_2" (sequence); and $\pi_1 \cup \pi_2$ means "either execute program π_1 or execute program π_2" (non-deterministic choice). LRC allows only a limited form of iteration, as follows. If π is a program that does not contain an element of $\mathcal{A}$ (i.e., all sub-programs of π are built from the basic reveal and conceal actions in RC), then π^* means "repeatedly execute π an undetermined number of times". For technical reasons, (discussed in more detail below), we choose to omit the standard dynamic logic "test" operator, $\varphi?$, from the logic LRC.

Finally, and again for somewhat technical reasons, we include within LRC a universal modality $\Box\varphi$. The expression $\Box\varphi$ means "in all states of the model, fixing the current visibility descriptions, φ holds".

2.2 Some Example Formulae

Let us see some examples of formulae of our logic.

$$p \rightarrow [r(p,i)]K_i p$$

This formula says that, if p is true, then after revealing the variable p to i, agent i will know that p is true. This is in fact a valid formula of LRC: the effect of revealing a Boolean variable to an agent will be that the agent knows the value of that Boolean variable.

$$[\alpha]p \rightarrow [r(p,i);\alpha]K_i p$$

This formula says that if after doing α, the variable p is true, then if we reveal p to i and then do α, then i will know that p is true. This is in fact also a valid valid formula of LRC, for all $\alpha \in \mathcal{A}$.

$$K_i r \wedge [c(p,i) \cup c(q,i)]\neg K_i r$$

This formula says that i knows r, but if we choose to conceal either p or q, then i will not know r.

$$\langle r(v,i)^*\rangle\varphi \leftrightarrow \varphi \vee \langle r(v,i)\rangle\varphi$$

This (valid) formula says that φ is true after some undetermined number of executions of the $r(v,i)$ action if, and only if, φ is true now or after at most one repetition of the action $r(v,i)$.

Note that the following is not a well-formed formula of LRC.

$$[(r(v,i) \cup (\alpha_1;\alpha_2))^*]\varphi$$

This is because the iteration operator " * " is here applied to a program that contains state changing actions, i.e., elements of $\mathcal{A}$. We impose this restriction on iteration mainly for technical reasons. Having such formulae would greatly complicate any completness proof for an axiomatic system for LRC, while hiding the main idea behind some difficult technical details; moreover, as is well known, a strong completeness proof is out of reach in this situation. Note also that we do not have programs of the form $\varphi?$. This is justified by the following reasoning. We want to formulate a logic that

π	$::=$	α	atomic state changing action
	$\mid$	$r(v,i)$	reveal v to i
	$\mid$	$c(v,i)$	conceal v from i
	$\mid$	$\pi;\pi$	sequence
	$\mid$	$\pi \cup \pi$	non-deterministic choice
	$\mid$	π^*	repeat π some finite number of times (π must contain no actions $\mathcal{A}$)
	$\mid$	*skip*	do nothing
φ	$::=$	$\top$	truth constant
	$\mid$	p	propositional atoms
	$\mid$	$sees_i(v)$	agent i sees variable v
	$\mid$	$\neg\varphi$	negation
	$\mid$	$\varphi \vee \varphi$	disjunction
	$\mid$	$K_i\varphi$	epistemic box modality
	$\mid$	$[\pi]\varphi$	dynamic box modality
	$\mid$	$\Box\varphi$	universal modality for fixed visibility structure

Figure 1: Syntax of programs (π) and formulae (φ). Terminal symbols are interpreted as follows: $\alpha \in \mathcal{A}$ is an atomic state changing action, $p \in \Phi \cup Vis$ is an arbitrary Boolean variable, $v \in Vis$ is a visibility variable, and $i \in \mathcal{N}$ is an agent.

can be used for reasoning about the knowledge that can be obtained *only* from directly revealing features of the environment. Since the agents "know" the program that is being executed, performing a test on the value of a variable that is not visible to a certain agent can increase the agent's knowledge. This increase, however, is not because of a direct observation of the variable.

We define the syntax of programs π and formulae φ of the logic LRC by mutual induction through the grammar in Figure 1.

2.3 Models

(The reader may benefit from reading this section together with Example 1, below.) In what follows, we will assume the sets $\mathcal{N}$, Φ, and *Vis* are fixed, with each respective set playing the role described above.

Now, as we explained earlier, every agent $i \in \mathcal{N}$ is assumed to be able to completely and correctly see a subset $V_i \subseteq Vis$. That is, agent i will *know the value of the variables in* V_i; if $p \notin V_i$, then i does not necessarily know the value of p. We refer to V_i as the *visibility set* for agent i, and we refer to a tuple $V = (V_1, \dots, V_n)$, in which we have one visibility set for each agent, as a *visibility structure*. Notice that we place no requirements on visibility sets V_i or visibility structures $(V_1, \dots, V_n)$. It could be that $V_i = V_j$, for example, or even that $V_1 = \dots = V_n = \emptyset$ (although this latter case would not be very interesting). Let $\mathcal{V}$ denote the set of visibility structures.

Next, we assume a set $\mathcal{S} = \{s_1, \dots, s_m\}$ of *states*, and a standard Kripke valuation function $\theta : \mathcal{S} \to 2^{\Phi \cup Vis}$, which gives the set of variables $\theta(s)$ true in each state $s \in \mathcal{S}$. Notice that θ gives a value both for variables in Φ and variables in *Vis*.

For each state changing action $\alpha \in \mathcal{A}$, we are assumed to have a binary relation $R_\alpha \subseteq S \times S$, capturing the effects of α. The interpretation of this relation is more or less standard for dynamic logic [7]: if $(s, s') \in R_\alpha$, then this means that state s' could result as a possible effect of performing action α in state s.

Putting these components together, a *model*, m, (over the sets fixed above) is a structure

$$m = \langle \mathcal{S}, V, \{R_\alpha\}, \mathcal{R}_\diamond, \theta \rangle,$$

where $\mathcal{S}$ is a state set, $V \in \mathcal{V}$ is a visibility structure, $\{R_\alpha\}$ is a collection of accessibility relations for the state changing actions in $\mathcal{A}$, $\mathcal{R}_\diamond$ is the universal relation on $\mathcal{S}$, and θ is a Kripke valuation function. Let $\mathcal{M}$ denote the set of models.

A *pointed model* is a pair (m, s), where $m \in \mathcal{M}$ is a model and s is a state in m. Below, we will define the satisfaction of formulae with respect to pointed models. Let $\mathcal{P}(\mathcal{M})$ be the set of all pointed models over the set of models $\mathcal{M}$. A *configuration* $f = \langle \mathcal{S}, \{R_\alpha\}, \mathcal{R}_\diamond, \theta \rangle$ abstracts away from the specific visibility structure V. The set M_f is the set of all models over the configuration f.

Where s and s' are two states in $\mathcal{S}$, we write $s \sim_i s'$ to mean that the states s and s' agree on the values of variables V_i, i.e.,

$$s \sim_i s' \qquad \text{iff} \qquad \theta(s) \cap V_i = \theta(s') \cap V_i.$$

The reader will note that the relations $\sim_i$ defined in this way are equivalence relations, and we will later use these relations to define a conventional (S5) interpretation for knowledge modalities, cf. [4].

2.4 Dynamic Accessibility Relations

We must now define the accessibility relations R_π, used to give a semantics to dynamic modalities $[\pi]$ (cf. [6, p.87]). In Propositional Dynamic Logic (PDL), program accessibility relations R_π are binary relations over the set $\mathcal{S}$ of system states. In our logic, they are slightly more complex: they are binary relations over the set $\mathcal{P}$ of pointed models (m, s). No actions in our framework will change a configuration: state changing actions (as the name suggests) change a state, while revealing and concealing actions *RC* change the visibility structure, and hence the model. We will define these relations in three stages: first we define the relations for atomic revelation and concealment programs $r(p, i)$ and $c(p, i)$, then we define the form of accessibility relations for state changing actions, and then finally, we define the accessibility relations for complex programs with respect to these.

Assume $m = (\mathcal{S}, V, \{R_\alpha\}, \mathcal{R}_\diamond, \theta)$ and $m' = (\mathcal{S}', V', \{R'_\alpha\}, \mathcal{R}'_\diamond, \theta')$ are models and s, s' are states such that $s \in \mathcal{S}$ and $s' \in \mathcal{S}'$. Then

$$((m, s), (m', s')) \in \mathcal{R}_{r(v,i)} \text{ iff:}$$

1. $s' = s, \mathcal{S}' = \mathcal{S}, \mathcal{R}' = \mathcal{R}, \mathcal{R}'_\diamond = \mathcal{R}_\diamond$ and $\theta' = \theta$
   Revealing v to i does not change the current point, the state set, any of the accessibility relations, or the truth of atomic propositions.
2. $V'_i = V_i \cup \{v\}$ and for all $j \neq i$, $V'_j = V_j$
   Atom v becomes visible for i after $r(v, i)$ has been executed, but otherwise, i's visibility set is unchanged and the visibility set of every other agent remains unchanged.

We define the relations $\mathcal{R}_{c(v,i)}$ in a similar way:

$$((m, s), (m', s')) \in \mathcal{R}_{c(v,i)} \text{ iff:}$$

1. $s' = s, \mathcal{S}' = \mathcal{S}, \mathcal{R}' = \mathcal{R}, \mathcal{R}'_\diamond = \mathcal{R}_\diamond$ and $\theta' = \theta$
   Concealing v from i does not change the current point, the state set, any of the accessibility relations, or the truth of atomic propositions.
2. $V'_i = V_i \setminus \{v\}$ and for all $j \neq i$, $V'_j = V_j$
   Atom v becomes invisible for i after $c(v, i)$ has been executed, but otherwise, i's visibility set is unchanged and the visibility set of every other agent remains unchanged.

The relation $\mathcal{R}_{skip}$ is the identity, i.e.,:

$$\mathcal{R}_{skip} = \{((m, s), (m, s))\}.$$

Let $\mathcal{R}_{r(p,i)^*} = (\mathcal{R}_{r(p,i)})^*$ and $\mathcal{R}_{c(p,i)^*} = (\mathcal{R}_{c(p,i)})^*$, i.e., $\mathcal{R}_{r(p,i)^*}$ is the reflexive and transitive closure of the relation $\mathcal{R}_{r(p,i)}$; similarly for $\mathcal{R}_{c(p,i)^*}$.

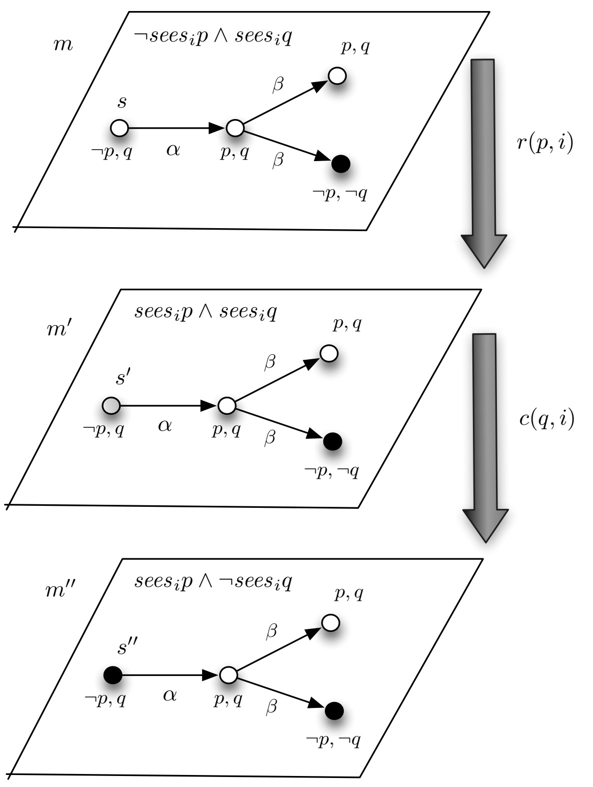


Figure 2: Three models.

EXAMPLE 1. *Consider the models m, m' and m'' from Figure 2, where $\mathcal{N} = \{i\}$ and $Vis = \{p, q\}$. The three models are connected by a $r(p,i)$ and a $c(q,i)$ transition, respectively. Note they are all models over the same configuration (i.e., the only difference between the models is the visibility structure). Points in a model in the same $\sim_i$-equivalence class have the same 'colour' (in m', the state s' induces the equivalence class $\{s'\}$). Since the visibility descriptions are global, they are given at the top of each model. We have the following (for all φ), where the reader may like to take a peek at the definition of entailment at the end of this section:*

1. $(m,s) \models K_i q \wedge \neg K_i p \wedge (\varphi \leftrightarrow [\alpha] K_i \varphi)$
The agent knows q but not p, and the action α does not change his knowledge.

2. $(m,s) \models \langle r(p,i) \rangle (K_i \neg p \wedge [\alpha] K_i p)$
After revealing p to i, the agent knows p is false, but also that p would become true after α.

3. $(m,s) \models \langle r(p,i); \alpha; c(q,i) \rangle K_i [\beta] K_i (p \leftrightarrow q)$
It is possible to reveal p to i, then do α and then conceal q from i so that afterwards, the agent knows that all executions of β lead to states where the agent is sure that p and q are equivalent.

Note that the $\Box$ and $\Diamond$ operators enable us to quantify over states that are present in a model m. We have for instance in (m,s) that $\Box(\neg sees_i(p) \wedge \Diamond[\alpha \cup \beta]\bot)$: what an agent sees is the same in each snapshot, and in m, there is a state where neither α nor β can be performed.

We will now prove that the iteration operator, *, can in fact be eliminated from programs in LRC. This simplifies the semantics of the language, and greatly simplifies the completeness proof we give later. First, we prove a proposition which shows how reveal or conceal actions $\gamma(v,i)$ can be eliminated, or moved "along" a sequence *RC*-actions.

PROPOSITION 1. *The following are true. Let $\gamma(v,i), \hat{\gamma}(v,i)$ be either $r(v,i)$ or $c(v,i)$, such that $\gamma(v,i) = r(v,i)$ iff $\hat{\gamma}(v,i) = c(v,i)$.*

1. $\mathcal{R}_{\gamma(v,i)^*} = R_{skip} \cup \mathcal{R}_{\gamma(v,i)}$;
2. $\mathcal{R}_{\gamma(v,i)} \circ \mathcal{R}_{\gamma(v,i)} = \mathcal{R}_{\gamma(v,i)}$;
3. $\mathcal{R}_{\gamma(v,i)} \circ \mathcal{R}_{\hat{\gamma}(v,i)} = \mathcal{R}_{\hat{\gamma}(v,i)}$;
4. $\mathcal{R}_{r(v,i)} \circ \mathcal{R}_{r(w,k)} = \mathcal{R}_{r(w,k)} \circ \mathcal{R}_{r(v,i)}$;
5. $\mathcal{R}_{c(v,i)} \circ \mathcal{R}_{c(w,k)} = \mathcal{R}_{c(w,k)} \circ \mathcal{R}_{c(v,i)}$;
6. $\mathcal{R}_{r(v,i)} \circ \mathcal{R}_{c(w,k)} = \mathcal{R}_{c(w,k)} \circ \mathcal{R}_{r(v,i)}$*, where $i \neq k$ or $v \neq w$.*
7. $\mathcal{R}_{\gamma(v,i)} \circ \mathcal{R}_{\alpha} = \mathcal{R}_{\alpha} \circ \mathcal{R}_{\gamma(v,i)}$*, where $\alpha \in \mathcal{A}$.*

PROOF. Follows immediately from the definition of the relations $R_{r(v,i)}, R_{c(v,i)}, R_{skip}$. $\Box$

Using the above properties, we prove that for any program built from atomic programs in *RC* only, the following is true.

COROLLARY 1. *Let $\vec{\alpha}_i$ denote a sequence of RC-programs α_{i_1}; $\alpha_{i_2}; \ldots; \alpha_{i_k}$. Let n sequences of RC-programs $\vec{\alpha}_1, \vec{\alpha}_2, \ldots, \vec{\alpha}_n$ be given, and define Σ as the set of sequences σ of RC-programs that are made by choosing an arbitrary number of sequences from $\vec{\alpha}_1, \vec{\alpha}_2, \ldots, \vec{\alpha}_n$ (each $\vec{\alpha}_i$ occurring at most once), and combining them in an arbitrary order using only the operator ";". So $\Sigma = \{skip, \vec{\alpha}_1, \ldots, \vec{\alpha}_n, (\vec{\alpha}_1; \vec{\alpha}_2), \ldots, (\vec{\alpha}_n; \vec{\alpha}_{n-1}), (\vec{\alpha}_1; \vec{\alpha}_2; \vec{\alpha}_3), \ldots, (\vec{\alpha}_n; \vec{\alpha}_{n-1}; \vec{\alpha}_{n-2}), (\vec{\alpha}_1; \vec{\alpha}_2; \ldots; \vec{\alpha}_n), \ldots, (\vec{\alpha}_n; \vec{\alpha}_2; \ldots; \vec{\alpha}_{n-1})\}$. Then*

$$(\vec{\alpha}_1 \cup \ldots \cup \vec{\alpha}_n)^* \quad = \quad \cup_{\sigma \in \Sigma} \sigma$$

PROOF. The statement is best understood via an example. We claim that if $\alpha, \beta \in RC$ then

$$(\alpha \cup \beta)^* = skip \cup \alpha \cup \beta \cup (\alpha; \beta) \cup (\beta; \alpha)$$

This follows from the fact that

$$\begin{aligned} (\alpha \cup \beta)^* \quad = \quad & skip \cup (\alpha \cup \beta) \cup (\alpha \cup \beta); (\alpha \cup \beta) \\ & \cup (\alpha \cup \beta); (\alpha \cup \beta); (\alpha \cup \beta) \ldots . \end{aligned}$$

Consider $(\alpha \cup \beta); (\alpha \cup \beta); (\alpha \cup \beta)$. It is equivalent to

$$\alpha; \alpha; \alpha \cup \alpha; \alpha; \beta \cup \ldots \cup \alpha; \beta; \beta \ldots \cup \beta; \beta; \beta$$

Using the equivalences from Proposition 1, we see that this is actually equivalent to

$$\alpha \cup (\alpha; \beta) \cup (\beta; \alpha) \cup \beta.$$

The proof of the general statement is similar. $\Box$

COROLLARY 2. *Every program is equivalent to a program without the operator *.*

PROOF. This follows from Proposition 1, together with the fact that we do not allow the star operator to be applied to programs containing elements from $\mathcal{A}$ and Corollary 1. $\Box$

Given the atomic relations R_α for state changing actions and $\mathcal{R}_{r(p,i)}$, $\mathcal{R}_{c(p,i)}$ for visibility actions, we obtain the accessibility relations $\mathcal{R}_\pi$ for arbitrary programs π as follows. Let the composition of arbitrary relations R_1 and R_2 be denoted by $R_1 \circ R_2$. Then the accessibility relations for complex programs are defined [7]:

$$\begin{array}{rcl}\mathcal{R}_\alpha & = & \{((m,s),(m,s')) \mid (s,s') \in R_\alpha \;\&\; m \in \mathcal{M}\} \\ \mathcal{R}_{\pi_1;\pi_2} & = & \mathcal{R}_{\pi_1} \circ \mathcal{R}_{\pi_2} \\ \mathcal{R}_{\pi_1 \cup \pi_2} & = & \mathcal{R}_{\pi_1} \cup \mathcal{R}_{\pi_2}\end{array}$$

At last we are ready to give the formal semantics for our logic. Assume that φ is a formula of our logic and that $(m,s) \in \mathcal{P}$ is a pointed structure. Let $\mathcal{M}$ be the class of all models. Then we write $\mathcal{M},(m,s) \models \varphi$ to mean that φ is true at (satisfied in) state s of m. Since $\mathcal{M}$ is fixed, we also write $(m,s) \models \varphi$ for this. The satisfaction relation "$\models$" is inductively defined by the following rules:

$(m,s) \models \top$

$(m,s) \models p$ iff $p \in \theta(s)$ (where $p \in \Phi$);

$(m,s) \models sees_i(v)$ iff $v \in V_i$ (where $v \in Vis$);

$(m,s) \models \neg\varphi$ iff not $(m,s) \models \varphi$

$(m,s) \models \varphi \vee \psi$ iff $(m,s) \models \varphi$ or $(m,s) \models \psi$

$(m,s) \models K_i\varphi$ iff $\forall s'$ with $s \sim_i s'$, we have $(m,s') \models \varphi$

$(m,s) \models \Box\varphi$ iff $\forall s'$ we have $(m,s') \models \varphi$

$(m,s) \models [\pi]\varphi$ iff $\forall (m',s')$ such that $((m,s),(m',s')) \in \mathcal{R}_\pi$ we have $(m',s') \models \varphi$.

We define the remaining connectives of classical logic ("$\bot$" – falsum, "$\wedge$" – and, "$\rightarrow$" – implies, "$\leftrightarrow$" – if, and only if), the diamond dual M_i ("maybe") of the epistemic modality K_i, the diamond dual $\langle\pi\rangle$ of the dynamic box modality, and the diamond dual $\Diamond$ of the universal modality $\Box$ can be defined as abbreviations in the expected way:

$$\begin{array}{rcl}\bot & \hat{=} & \neg\top \\ \varphi \wedge \psi & \hat{=} & \neg(\neg\varphi \vee \neg\psi) \\ \varphi \rightarrow \psi & \hat{=} & (\neg\varphi) \vee \psi \\ \varphi \leftrightarrow \psi & \hat{=} & (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi) \\ M_i\varphi & \hat{=} & \neg K_i \neg\varphi \\ \langle\pi\rangle\varphi & \hat{=} & \neg[\pi]\neg\varphi. \\ \Diamond\varphi & \hat{=} & \neg\Box\neg\varphi.\end{array}$$

2.5 A Detailed Example

EXAMPLE 2. *Suppose we have three members of a committee, a, b, and c who are going to vote to elect a new committee chair. The standing chair is c and the new chair will be chosen from the three of them, by the three of them. Let $p^i_j \in \Phi$ denote that agent i's vote is for agent j (p^i_j is read as "i prefers j"', or "i votes for j"). We assume that votes are fair, in the sense that every agent votes exactly for one agent:*

$$\mu : \bigwedge_{i \in \mathcal{N}} \bigwedge_{k \neq j \neq m \neq k} p^i_j \leftrightarrow \neg p^i_k \wedge \neg p^i_m$$

The rule used for electing a winner is as follows: any agent will be elected if it has a majority of the votes; if there is no majority (everybody gets one vote each), then the standing chair, c, is elected.

Given this rule, let us define abbreviations w_i, denoting that i is the winner:

$$\begin{array}{rcl}w_a & \hat{=} & (\bigvee_{i,j \in \mathcal{N}: i \neq j}(p^i_a \wedge p^j_a)) \\ w_b & \hat{=} & (\bigvee_{i,j \in \mathcal{N}: i \neq j}(p^i_b \wedge p^j_b)) \\ w_c & \hat{=} & (\neg w_a \wedge \neg w_b)\end{array}$$

Let ω collect these three definitions as a conjunction. Finally, we specify who initially sees what: agents initially see only their own vote.

$$\sigma : \bigwedge_{i,j \in \mathcal{N}} sees(p^i_j, i) \wedge \bigwedge_{k \neq i, i,j,k \in \mathcal{N}} \neg sees(p^k_j, i)$$

To express the background information in this scenario, we find it convenient to define a common knowledge *operator, C. This is a standard construction, and we refer the reader to, e.g., [4] for details. Formally, given the individual knowledge operators K_i, we first define an "everyone knows" operator, E, as follows: $E\varphi \hat{=} \bigwedge_{i \in \mathcal{N}} K_i\varphi$. We then define the common knowledge operator $C\varphi$ as the maximal fixed point solution to the expression $E(\varphi \wedge C\varphi)$.*[1]

Now, let the background information χ be $\mu \wedge \omega \wedge \sigma$. We will assume that χ is common knowledge among $\{a, b, c\}$: all agents know (and they know that they know, etc) that they vote for only one candidate, what the definition of winning is, and which variables are initially seen.

Let us consider the vote $\nu = p^a_b \wedge p^b_c \wedge p^c_b$, which of course is not commonly known. We then have

$$(C\chi \wedge \nu) \rightarrow [r(p^a_b, c)]K_c w_b \tag{1}$$

If a's vote is revealed to c, then c knows who the winner is (it is b).

Agents a and b do not know that c in this case knows who the winner is, even if they know that c learns b's vote:

$$(C\chi \wedge \nu) \not\rightarrow (K_a[r(p^a_b, c)]K_a K_c w_b \vee K_b[r(p^a_b, c)]K_b K_c w_b) \tag{2}$$

Likewise, we have

$$(C\chi \wedge \nu) \rightarrow [r(p^b_c, a) \cup r(p^c_b, a)]K_a \neg w_a \tag{3}$$

If a learns the vote of one of the other committee members, he knows that he has not won the election.

Let now α be $(r(p^b_c, a) \cup r(p^c_b, a)); (c(p^b_c, a) \cup c(p^c_b, a))$ (randomly reveal one of the variables p^b_c and p^c_b to a, and then conceal randomly one of them). Then

$$(C\chi \wedge \nu) \rightarrow (\langle\alpha;\alpha\rangle K_a p^b_c \wedge \langle\alpha;\alpha\rangle \neg K_a p^b_c) \tag{4}$$

That is, there is a choice of revealing and concealing the variables so that a finds out he is winning, and there is a choice that he is not. This should be contrasted with the program $\alpha' = (r(p^b_c, a); c(p^b_c, a)) \cup (r(p^c_b, a); c(p^c_b, a))$, for which we obtain

$$(C\chi \wedge \nu) \rightarrow [\alpha';\alpha']\neg K_a p^b_c \tag{5}$$

(If the agent gets a random variable revealed and then concealed, he does not learn anything).

Our example easily illustrates how agents can know atoms without seeing them:

$$(C\chi \wedge \nu) \rightarrow \neg K_c \neg p^a_c \wedge [r(p^a_b, c)](\neg sees(p^a_c, c) \wedge K_c \neg p^a_c) \tag{6}$$

In words, given the initial constraints $(C\chi \wedge \nu)$, agent c does not know that p^a_c is false. However, after p^a_b is revealed to c, although he

[1]In the following, the C-operator does not appear in the consequent of any of the example formula, and hence can be omitted: for this example it suffices to think of it as an abbreviation of mutual knowledge of depth three.

still does not see p^a_c, he now knows its true value! This is because c knows that a only votes for one agent, i.e., he knows that certain constraints between variables exist.

When building or verifying intelligent agents, one might wonder what the benefit is of having conceal-actions, in which agents just 'forget' the truth value of certain atomic propositions. This makes sense in many scenario's where agents need at some time sufficient, maybe sensitive information to take an appropriate decision, where they should not 'accumulate' too much of this information. An example of this might be an agent who grants users access to a sensitive website, and the users posess an n-character password, where for each login session, they only are required to reveal k positions of this password ($k < n$). The agent deciding whether the user is allowed access should in such cases at every login attempt know whether the user has provided the right k characters, but when the login session ends, it should be ensured that this information is not remembered, since otherwise he in the end would learn the complete password. Instead of formalising this additional scenario, let us now show how, even in the voting setting, the possibility of concealing can be useful.

EXAMPLE 3. *Continuing with the voting example, let us introduce new atoms δ_i ($i \in \mathcal{N}$) meaning that i is the declared winner of the vote. Initially, we have*

$$\delta : \bigwedge_{i,j\in\mathcal{N}} \neg\delta_i \wedge sees(\delta_i, j) \tag{7}$$

That is, initially nobody is the declared winner, and the fact who is a declared winner is visible for each agent. We also assume an action β_i, which models that the chair c can declare that i is the winner. In order for i being enabled to be declared the winner, the pre-condition is $K_c w_i$ and the post-condition is δ_i.

$$(\neg K_c w_i \rightarrow \neg\langle\beta_i\rangle\top) \wedge (K_c w_i \rightarrow \langle\beta_i\rangle\delta_i) \tag{8}$$

Before defining our procedure for declaring the winner, let r_c (reveal to c) be short for

$$r_c : r(p^a_a, c); r(p^a_b, c); r(p^a_c, c); r(p^b_a, c); r(p^b_b, c); r(p^b_c, c)$$

Similarly, c_c is like r_c, but rather than revealing a's and b's votes to c, they are concealed from them. Define

$$\gamma = r_c; (\beta_a \cup \beta_b \cup \beta_c); c_c$$

Then γ has the following properties:

1. $C(\chi \wedge \delta) \rightarrow [\gamma](w_i \leftrightarrow K_a\delta_i \wedge K_b\delta_i \wedge K_c\delta_i)$
   After γ, any winner is known to be a declared winner

2. $C(\chi \wedge \delta) \rightarrow [\gamma]\neg(K_c p^a_a \vee K_c p^a_b \vee K_c p^a_c)$
   That is, after every execution of γ, agent c does not know (does not remember) a's vote (the same is of course true for b's vote). It is in fact easy to see that after execution of γ, we have $\neg K_j p^i_k$, for any $i \neq j$.

3. AXIOMS

We now present an axiomatization for LRC: the main technical result of our paper is that this axiomatization is sound and complete. We first present the axiomatization and discuss the properties the various axioms are capturing, before describing the completeness proof in the following subsection. The axiomatization is presented in Tables 1 and 2. Table 1 deals with the knowledge axioms, the axioms for $\Box$ and the inference rules of the logic. This is all fairly standard: the axioms say that both K_i and $\Box$ are $S5$-operators,

Propositional Logic:	
(Ax1)	propositional tautologies
S5 Axioms for Knowledge:	
(Ax2)	$K_i(\varphi \rightarrow \psi) \rightarrow (K_i\varphi \rightarrow K_i\psi)$
(Ax3)	$K_i\varphi \rightarrow \varphi$
(Ax4)	$\neg K_i\varphi \rightarrow K_i\neg K_i\varphi$
S5 Axioms for State of Revelation:	
(Ax5)	$\Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$
(Ax6)	$\Box\varphi \rightarrow \varphi$
(Ax7)	$\neg\Box\varphi \rightarrow \Box\neg\Box\varphi$
Inference Rules:	
(IR1)	From $\vdash \varphi \rightarrow \psi$ and $\vdash \varphi$ infer $\vdash \psi$
(IR2)	From $\vdash \varphi$ infer $\vdash K_i\varphi$
(IR3)	From $\vdash \varphi$ infer $\vdash [\pi]\varphi$

Table 1: Inference rules for LRC and some aixoms.

which is standard for knowledge (see e.g., [4]) and for the universal modality (see, e.g., [2]). Notice that the positive introspection axiom, ($K_i \rightarrow K_iK_i\varphi$), follows from the other axioms, and similarly for the $\Box$ modality.

The axioms of Table 2 relate to the dynamic component and the interaction between our modalities. Of the dynamic logic axioms:

(Ax8) and (Ax9) say that actions conceal and reveal are deterministic: they lead to a unique outcome.

(Ax10) and (Ax11) say that reveal and conceal are idempotent: repeating them has no effect.

(Ax12) and (Ax13) explain that when doing a reveal and a conceal action in sequence, it is the last performed action that determines the result.

(Ax14), (Ax15) and (Ax18) say that two actions from RC commute with each other, as long as they concern different agents or different variables.

According to (Ax16) and (Ax17), atomic actions from RC and from $\mathcal{A}$ commute. Semantically, this is illustrated by Figure 2: "horizontal" and "vertical" steps can be taken in arbitrary order.

(Ax19)–(Ax21) are a standard set of axioms for the dynamic logic constructs of our logic, with a direct correspondence to PDL [7, p.173].

Finally, we have 16 interaction axioms in LRC:

(Ax22) and (Ax23) 'generalise' axioms (Ax16) and (Ax17), and are again illustrated by Figure 2.

Axioms (Ax24)–(Ax27) are *persistence properties* of *sees* and $\neg$*sees*. (Ax24) and (Ax25) for instance say that the fact that agent i sees variable v is not undone by concealing either another variable from i, or concealing a variable from another agent (Ax24), and persists through any reveal action.

(Ax28) and (Ax29) say that who sees what is a global property in a model m (the only way to change this is to perform an RC-action, which leads to a model m').

(Ax30) and (Ax31) give the non-persistence of *sees* (it can become false through an apropriate conceal action) and $\neg$*sees* (which can become false through an appropriate reveal action).

Dynamic Logic Axioms:		
(Ax8)	$\langle c(v,i)\rangle\varphi \leftrightarrow [c(v,i)]\varphi$	
(Ax9)	$\langle r(v,i)\rangle\varphi \leftrightarrow [r(v,i)]\varphi$	
(Ax10)	$\langle r(v,i);r(v,i)\rangle\varphi \leftrightarrow \langle r(v,i)\rangle\varphi$	
(Ax11)	$\langle c(v,i);c(v,i)\rangle\varphi \leftrightarrow \langle c(v,i)\rangle\varphi$	
(Ax12)	$\langle r(v,i);c(v,i)\rangle\varphi \leftrightarrow \langle c(v,i)\rangle\varphi$	
(Ax13)	$\langle c(v,i);r(v,i)\rangle\varphi \leftrightarrow \langle r(v,i)\rangle\varphi$	
(Ax14)	$\langle c(v,i);c(w,k)\rangle\varphi \leftrightarrow \langle c(w,k);c(v,i)\rangle\varphi$	if C_1
(Ax15)	$\langle r(v,i);r(w,k)\rangle\varphi \leftrightarrow \langle r(w,k);r(v,i)\rangle\varphi$	if C_1
(Ax16)	$\langle c(v,i)\rangle\langle\alpha\rangle\varphi \leftrightarrow \langle\alpha\rangle\langle c(v,i)\rangle\varphi$	if C_2
(Ax17)	$\langle r(v,i)\rangle\langle\alpha\rangle\varphi \leftrightarrow \langle\alpha\rangle\langle r(v,i)\rangle\varphi$	if C_2
(Ax18)	$\langle r(v,i);c(w,k)\rangle\varphi \leftrightarrow \langle c(w,k);r(v,i)\rangle\varphi$	if C_1
(Ax19)	$[\pi](\varphi \to \psi) \to ([\pi]\varphi \to [\pi]\psi)$	
(Ax20)	$[\pi_1 \cup \pi_2]\varphi \leftrightarrow [\pi_1]\varphi \wedge [\pi_2]\varphi$	
(Ax21)	$[\pi_1;\pi_2]\varphi \leftrightarrow [\pi_1][\pi_2]\varphi$	
Interaction Axioms:		
(Ax22)	$\Diamond\langle r(v,i)\rangle\varphi \leftrightarrow \langle r(v,i)\rangle\Diamond\varphi$	
(Ax23)	$\Diamond\langle c(v,i)\rangle\varphi \leftrightarrow \langle c(v,i)\rangle\Diamond\varphi$	
(Ax24)	$sees_i(v) \to [c(w,k)]sees_i(v)$	if C_1
(Ax25)	$sees_i(v) \to [r(w,j)]sees_i(v)$	
(Ax26)	$\neg sees_i(v) \to [c(w,j)]\neg sees_i(v)$	
(Ax27)	$\neg sees_i(v) \to [r(w,k)]\neg sees_i(v)$	if C_1
(Ax28)	$sees_i(v) \to \Box sees_i(v)$	
(Ax29)	$\neg sees_i(v) \to \Box\neg sees_i(v)$	
(Ax30)	$sees_i(v) \to \langle c(v,i)\rangle\neg sees_i(v)$	
(Ax31)	$\neg sees_i(v) \to \langle r(v,i)\rangle sees_i(v)$	
(Ax32)	$\varphi_0 \to [\pi]\varphi$	if C_3
(Ax33)	$\Box\varphi \to [\alpha]\varphi$	if C_2
(Ax34)	$\Box\varphi \to K_i\varphi$	
(Ax35)	$sees_i(v) \wedge v \to K_i v$	
(Ax36)	$sees_i(v) \wedge \neg v \to K_i\neg v$	
(Ax37)	$\big(\bigwedge_{u\in U}(u \wedge sees_i(u)) \wedge \bigwedge_{v\in V}(\neg v \wedge sees_i(v)) \wedge \bigwedge_{w\in W}\neg sees_i(w)\big) \to (K_i\varphi \to \Box((\bigwedge_{u\in U} u \wedge \bigwedge_{v\in V}\neg v) \to \varphi))$	if C_4

Table 2: An axiomatization. The condition C_1 reads $i \neq k$ or $v \neq w$, condition C_2 is that $\alpha \in \mathcal{A}$, condition C_3 is φ_0 is objective and π contains no actions from $\mathcal{A}$, and, finally, C_4 is that $Vis = U \cup V \cup W$.

(Ax32) says that actions from RC do not change the value of atoms in a sate.

(Ax33) and (Ax34) explain that any action from $\mathcal{A}$ keeps us in m and an agent only considers states in m possible.

(Ax35) and (Ax36) capture the basic interaction between visibility sets and knowledge: it an agent i sees a variable v, then i correctly knows the value of v Notice that the converse implication does not hold: to see this, suppose the state set $\mathcal{S}$ was a singleton; then every agent would know the value of every variable, irrespective of whether they could see it or not.

Finally, (Ax37) splits up the visibility atoms in three sets U, V and W. The atoms in U are all true and seen by i. The atoms in V are false and seen by i. None of the atoms in W are seen by i. Then, if i knows that φ, then φ must be true in every state that agrees on the atoms in U and V. In other words, every state (in the same m) that agrees on the atoms that i sees is considered possible by him.

3.1 Completeness

The technical details of our completeness proof are rather involved, and so here we will simply describe the key steps on which these details are based. By Corollary 2, we can restrict ourselves to the language without the operator *. We work with the canonical model for our logic, which is built from maximal consistent sets (see [2] for the relevant notions): completeness of our logic follows from Lemma 2, below. In a nutshell, we take a consistent formula, include it in a maximal consistent set s, and in the canonical model, where states are consistent sets, truth in the model at state s and membership of a formula in s coincide, demonstrating that consistent formulas. The fact that we have eliminated the star operator from our language means that we do not have to introduce the machinery of the Fisher-Ladner closure that is used to deal with the non-compactness of the full Propositional Dynamic Logic. This, however does not mean that a completeness proof will be straightforward, because we have a technical problem of a different nature. In particular, the canonical model for our logic consists of maximal consistent sets of formulae (MCS) that are related via the canonical relations $R^c_\alpha, R^c_\Diamond, R^c_i$ in the usual way. Of course, we can prove a truth lemma with respect to this model but the model itself does not consist of pointed models of the form (m,s) that are related in the desired way. Therefore, this canonical model must be transformed so that it has the desired properties.

DEFINITION 1. *The canonical model for our logic is*

$$M = \langle W, \mathcal{R}^c_\Diamond, \mathcal{R}^c_\alpha, \mathcal{R}^c_i, \theta^c\rangle, \quad \textit{where:}$$

- *W is the set of all maximal consistent sets of formulae;*
- *$\Gamma R^c_\alpha \Delta$ iff for all formulae φ: if $\varphi \in \Delta$, then $\langle\alpha\rangle\varphi \in \Gamma$;*
- *$\Gamma R^c_\Diamond \Delta$ iff for all formulae φ: if $\varphi \in \Delta$, then $\Diamond\varphi \in \Gamma$;*
- *$\Gamma R^c_i \Delta$ iff for all formulae φ: if $\varphi \in \Delta$, then $M_i\varphi \in \Gamma$;*
- *$\theta^c(p) = \{\Gamma \in W \mid p \in \Gamma\}$.*

It is a standard result in modal logic that the relations $R^c_\Diamond$ and R^c_i are reflexive symmetric and transitive. In addition, the axioms governing composition and union of programs are Sahlqvist formulae and, therefore, canonical. It is a standard exercise in modal logic to prove that R^c_α satisfies the regularity conditions:

- $R^c_{\alpha;\beta} = R^c_\alpha \circ R^c_\beta$
- $R^c_{\alpha\cup\beta} = R^c_\alpha \cup R^c_\beta$.

We begin the transformation of this canonical model by defining pointed models of the form (m,s).

DEFINITION 2. *For every MCS $s \in W$ and all $\alpha \in \mathcal{A}$, $mod(s) = (m,s) = \langle S, V, R_\alpha, R_\Diamond, \theta\rangle$, where*

1. *$S = \{t \in W \mid sR^c_\Diamond t\}$;*
2. *$R_\alpha = R^c_\alpha \cap (S \times S)$;*
3. *$R_\Diamond = R^c_\Diamond \cap (S \times S)$;*
4. *$V = \langle V_1, \ldots, V_n\rangle$ is such that $v \in V_i$ iff $sees_i(v) \in s$;*
5. *$\theta(p) = \theta^c(p) \cap (S \times S)$.*

It follows from 1 that $R_\Diamond$ is the universal relation on S. We could also add an epistemic relation $R_i = R^c_i \cap (S \times S)$ to interpret knowledge: (Ax33) and (Ax34) ensure that $R_i \subseteq R_\Diamond$ and $R_\alpha \subseteq R_\Diamond$. Of course, we have that R_i and $R_\Diamond$ are equivalence relations because reflexivity, transitivity and symmetry are modally definable universal properties that are preserved under taking generated sub-models. We need to show that in fact this R_i relation captures exactly the truth-definition of knowledge in models based on the relation $\sim_i$: in fact, (Ax37), together with the axioms that relate knowledge with *sees* and $\Box$ are crucial here. In particular, Ax37 guarantees that within a fixed visibility structure, given a state s, there are no states s' for which $s \sim_i s'$ yet agent i would not consider them the same.

LEMMA 1 (EXISTENCE LEMMA FOR POINTED MODELS). *For all pointed models (m, s) as defined above and all $t \in S$,*

$\Diamond\varphi \in t$ iff there is a $t_1 \in S$ such that $tR_\Diamond t_1$ and $\varphi \in t_1$;

$M_i\varphi \in t$ iff $\exists t_1 \in S$ such that $tR_i t_1$ and $\varphi \in t_1$;

$R_\alpha\varphi \in t$ iff $\exists t_1 \in S$ such that $tR_\alpha t_1$ and $\varphi \in t_1$.

PROOF. By induction on the structure of φ. $\Box$

Having defined our pointed models. We move to defining our transformed canonical model. We start with defining the following operation:

DEFINITION 3. *If $(m, s) = \langle S, V, R_\alpha, R_\Diamond, \theta\rangle$, then*

$r(v, i)(m, s) = mod(s')$ as defined in Definition 2, where $s' = s \setminus \{\neg sees_i(v)\} \cup \{sees_i(v)\}$

$c(v, i)(m, s) = mod(s')$ as defined in Definition 2, where $s' = s \setminus \{sees_i(v)\} \cup \{\neg sees_i(v)\}$

Intuitively, we form the set $r(v, i)(m, s)$ by collecting all maximal consistent sets that are related via the canonical relation $R^c_{r(v,i)}$ to the maximal consistent sets in S.

PROPOSITION 2. *For all pointed models (m, s), $r(v, i)(m, s)$ and $c(v, i)(m, s)$ are pointed models.*

We now define a structure that collects all pointed models.

DEFINITION 4. *$\mathbb{M} = \{\mathbb{W}, \mathbb{R}_\pi\}$, where*

- *$\mathbb{W} = \{(m, s) \mid s \in W\}$, i.e., $\mathbb{W}$ consists of all pointed models for every MCS in the canonical model M as defined in Definition 1.*
- *$\mathbb{R}_\pi$ is defined inductively as follows.*
  1. *If π is an atomic state changing action $\alpha \in \mathcal{A}$, then $(m, s)\mathbb{R}_\pi(m_1, s_1)$ iff $m = m_1$ and $sR_\pi s_1$ in the sense of item 2 from Definition 2.*
  2. *If π is $r(v, i)$, then*
  $$(m, s)\mathbb{R}_\pi(m_1, s_1) \text{ iff } (m_1, s_1) = r(v, i)(m, s);$$
  3. *If π is $c(v, i)$, then*
  $$(m, s)\mathbb{R}_\pi(m_1, s_1) \text{ iff } (m_1, s_1) = c(v, i)(m, s);$$
  4. *$\mathbb{R}_{\alpha_1;\alpha_2} = \mathbb{R}_{\alpha_1} \circ \mathbb{R}_{\alpha_1}$;*
  5. *$\mathbb{R}_{\alpha_1\cup\alpha_2} = \mathbb{R}_{\alpha_1} \cup \mathbb{R}_{\alpha_1}$.*

Now we can prove the desired Truth lemma

LEMMA 2 (TRUTH LEMMA FOR.). *For all LRC formulae φ:*

$$\mathbb{M}, (m, s) \models \varphi \text{ iff } \varphi \in s.$$

Completeness then follows via a standard argument. Note that the semantic structure $\mathbb{M}$, (m, s) 'corresponds' to the set of all possible pointed models.

4. CONCLUSIONS

We developed a logic LRC, that allows us to reason about the effects of epistemic actions that reveal and conceal parts of an environment to an agent. In that sense, our logic is a direct 'dynamisation' of the interpreted systems approach to epistemic logic. Such epistemic actions seem very natural, and we believe that several applications of our logic are possible, for example in the area of security (LRC might for instance model situations where a user can access a secure website by only revealing part of his password). For future work, it will be interesting to consider the possibility of revealing and concealing *actions*, rather than variables, although this is likely to require a more elaborate semantic framework than that presented in this paper. Another natural extension would be to weaken the assumption that it is publicly known who sees which variables.

One of the few papers we are aware of that deal with ontic and epistemic actions is [12]. There, the state changing actions are assignments, and the epistemic actions are public announcements. In our framework, we can model public announcements of atoms (just reveal the value to all agents), but not directly disjunctions of them for instance, or epistemic formulas. On the other hand, [12] does not offer a logic for their framework: perhaps a synergy between their and our framework would lead to an interesting and well-understood framework that mixes ontic and epistemic actions. One might also expect that our approach is related to work on *awareness*, especially dynamic versions of it (cf. [1]). Although this warrants further investigation, a main difference is that if an agent i does not see p, we still have $K_i(p \vee \neg p)$ (the agent knows that p has some truth value), whereas, if i is not aware of p, the negation of this holds.

Interesting venues for further research are the connection with Dynamic (Epistemic) Logic, and decidability. As shown in [8], having a 'universal modality' may jeopardise decidability of a logic, but since our universal modality is 'restricted' to visibility structures, the logic might well be decidable.

5. REFERENCES

[1] J. van Benthem and F. R. Velázquez-Quesada. The Dynamics of Awareness. *Synthese*, **177**:1, pp. 5–27, 2010.
[2] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge UP, 2001.
[3] H. Ditmarsch, W. Hoek, and B. Kooi. *Dynamic Epistemic Logic*. Springer, 2007.
[4] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.
[5] J. Gerbrandy. Logics of propositional control. In *Proc. AAMAS-2006*, Hakodate, Japan, 2006.
[6] R. Goldblatt. *Logics of Time and Computation*. CSLI, 1987.
[7] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.
[8] E. Hemaspaandra. The Price of Universality. *Notre Dame J. Formal Logic* **37**:2, pp. 174–203, 1996.
[9] C. Lutz. Complexity and succinctness of public announcement logic. In *Proc. AAMAS-2006*, Hakodate, Japan, 2006.
[10] W. van der Hoek, D. Walther, and M. Wooldridge. Reasoning about the transfer of control. *JAIR*, 37:437–477, 2010.
[11] W. van der Hoek and M. Wooldridge. On the logic of cooperation and propositional control. *Artificial Intelligence*, 64:81–119, 2005.
[12] H. van Ditmarsch, W. van der Hoek, and B. Kooi. Dynamic epistemic logic with assignment. *Proc. AAMAS-2005*, pages 141–148, New York, USA, 2005. ACM Inc.

State and Path Coalition Effectivity Models for Logics of Multi-Player Games

Valentin Goranko
Informatics and Mathematical Modelling,
Technical University of Denmark
and University of Johannesburg
vfgo@imm.dtu.dk

Wojciech Jamroga
Computer Science and Communication,
University of Luxembourg
wojtek.jamroga@uni.lu

ABSTRACT

We consider models of multi-player games where abilities of players and coalitions are defined in terms of sets of outcomes which they can effectively enforce. We extend the well studied state effectivity models of one-step games in two different ways. On the one hand, we develop multiple state effectivity functions associated with different long-term temporal operators. On the other hand, we define and study coalitional path effectivity models where the outcomes of strategic plays are infinite paths. For both extensions we obtain representation results with respect to concrete models arising from concurrent game structures. We also apply state and path coalitional effectivity models to provide alternative, arguably more natural and elegant semantics to the alternating-time temporal logic ATL*, and discuss their technical and conceptual advantages.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Modal logic*; J.4 [**Social and Behavioral Sciences**]: Economics

General Terms

Theory

Keywords

Games models, effectivity, strategic logic

1. INTRODUCTION

A wide variety of multi-player games can be modeled by so called 'multi-player game models' a.k.a. 'concurrent game structures' [9, 3] which can be seen as a generalization of extensive form games or of repeated normal form (strategic) games. Here, we view them as general models of (qualitative) *multi-step games*. Intuitively, such game is based on a labelled transition system where every state is associated with a normal form game and the transitions between states are labelled by tuples of actions,[1] one for each player. Thus, the outcome of playing a normal form game at a given state is a transition to a new state, respectively to a new normal form game. In the quantitative version of such games, the outcome states are also associated with payoff vectors, while in the version that we consider here, the payoffs are *qualitative* – defined by properties of the outcome states, possibly expressed in a logical language. The players' objectives in multi-step games can simply be about reaching a desired ('winning') state, or they can be more involved, such as forcing a desired *long-term behaviour* (transition path, run).

Various logics for reasoning about coalitional abilities in multi-player games have been proposed and studied in the last two decades – most notably, Coalition Logic (CL) [9] and Alternating-time Temporal Logic (ATL* and its fragment ATL) [3]. Coalition Logic can be seen as a logic for reasoning about abilities of coalitions in one-step (strategic) games to bring about an outcome *state* with desired properties by means of single actions, while ATL* allows to express statements about multi-step scenarios. For example, the ATL formula $\langle\!\langle C\rangle\!\rangle F\varphi$ says that the coalition of players (or agents) C can ensure that φ will become true at *some* future moment, no matter what the other players do; likewise, $\langle\!\langle C\rangle\!\rangle G\varphi$ expresses that the coalition C can enforce φ to be *always* the case. More generally, the ATL* formula $\langle\!\langle C\rangle\!\rangle\gamma$ holds true iff C has a strategy to ensure that any resulting behavior of the system (i.e., any play of the game) will satisfy the temporal property γ.

In this paper we study how multi-step games can be modeled and characterized in terms of *effectivity of coalitions* with respect to possible outcome states or behaviours, and how such models can be used to provide conceptually simple and technically elegant semantics for logics of multi-player games such as ATL*. The paper has three main objectives:

(i) To extend the semantics for CL based on one-step coalitional effectivity to semantics for ATL over state-based coalitional effectivity models;
(ii) To develop the analogous notion of *coalitional path effectivity* representing the powers of coalitions in multi-step games to ensure long-term behaviors, and to provide semantics for ATL* based on it;
(iii) To obtain characterizations of multi-player game models in terms of abstract state and path coalitional effectivity models, analogous to Pauly's representation theorem [9, 5].

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

[1]Such actions are also called 'strategies' in normal form games, but we reserve the use of the term 'strategy' for a *global conditional plan* in a multi-step scenario.

We argue that characterizing effectivity of coalitions in multi-step games in terms of paths (cf. points (ii) and (iii) above) is conceptually more natural and elegant than in terms of outcome states, in several respects. First, collective strategies in such games generate *outcome paths* (plays), not just outcome states. Second, *one* path effectivity function is sufficient to define the powers of coalitions in a multi-step game for all kinds of temporal patterns, through the standard semantics of temporal operators. This point is further supported by the fact that path effectivity models provide a straightforward semantics for the whole language of ATL* (which is not definable by alternation-free fixpoint operators on the one-step ability). Finally, we argue that path effectivity can just as well be applied to variants of ATL(*) with imperfect information, where even simple modalities do not have fixpoint characterizations [6]. Still, also in that case, executing a strategy 'cuts out' a set of possible paths, just like in the perfect information case.

The paper is structured as follows. We begin by introducing basic notions in Section 2. In Section 3 we develop state-based effectivity models that suffice to define semantics of ATL. The models include three different effectivity functions, one for each basic modality $X, G, \mathcal{U}$. Then, in Section 4 we develop and study effectivity models based on paths. We show how they provide semantics to ATL*, and identify appropriate "playability" conditions, which we use to establish correspondences between powers of coalitions in the abstract models and strategic abilities of coalitions in concurrent game models. Finally, we briefly discuss how the path-oriented view can be used to facilitate reasoning about games with imperfect information in Section 5.

2. PRELIMINARIES

We begin by introducing some basic game-theoretic and logical notions. In all definitions hereafter, the sets of players, game (outcome) states, and actions available to players are assumed non-empty. Moreover, the set of players is always assumed finite.

2.1 Concurrent game models

Strategic games (a.k.a. normal form games) are basic models of non-cooperative game theory [8]. Following the tradition in the qualitative study of games we focus on abstract game modes, where the effect of strategic interaction between players is represented by abstract outcomes from a given set and players' preferences are not specified.

Definition 1 (Strategic game). *A* strategic game *is a tuple $G = (\mathbb{A}\text{gt}, St, \{Act_i | i \in \mathbb{A}\text{gt}\}, o)$ consisting of a set of players (agents) $\mathbb{A}\text{gt}$, a set of outcome states St, a set of actions (atomic strategies) Act_i for each player $i \in \mathbb{A}\text{gt}$, and an outcome function $o : \prod_{i\in\mathbb{A}\text{gt}} Act_i \to St$ which associates an outcome with every action profile.*

We define coalitional strategies α_C in G as tuples of individual strategies α_i for $i \in C$, i.e., $Act_C = \prod_{i\in C} Act_i$.

Strategic games are one-step encounters. They can be generalized to multi-step scenarios, in which every state is associated with a strategic game. Such games are also known as *concurrent game structures* [3].

Definition 2 (Concurrent game structures). *A* concurrent game structure *(CGS) is a tuple*

$$F = (\mathbb{A}\text{gt}, St, Act, d, o)$$

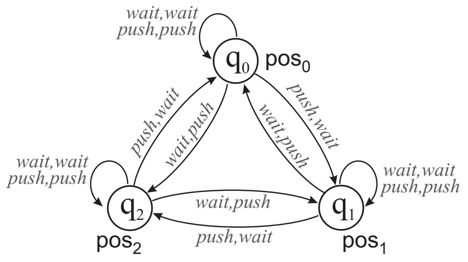


Figure 1: Repeated matching pennies: a concurrent game model M_1.

which consists of a set of players $\mathbb{A}\text{gt} = \{1, \dots, k\}$, a set of states St, a set of (atomic) actions Act, a function $d : \mathbb{A}\text{gt} \times St \to \mathcal{P}(Act)$ that assigns a sets of actions available to players at each state, and a (deterministic) transition function o that assigns the outcome state $o(q, \alpha_1, \dots, \alpha_k)$ to every starting state q and a tuple of actions $\langle\alpha_1, \dots, \alpha_k\rangle$, $\alpha_i \in d(i, q)$, that can be executed by $\mathbb{A}\text{gt}$ in q.

A concurrent game model *(CGM) is a CGS endowed with a valuation $V : St \to \mathcal{P}(Prop)$ for some fixed set of atomic propositions $Prop$.*

Note that in a CGS all players execute their actions synchronously and the combination of the actions, together with the current state, determines the transition in the CGS.

Example 1 (Repeated matching pennies). *Two agents play matching pennies repeatedly on a triangular board in such a way that the initial state of the next game depends on what they did before. More precisely, showing the heads means that the player wants to push the token, and showing the tails means that she wants the token to be left in the same place. Moreover, player 1 can only push the token to the right, while 2 can only push it to the left. The scenario can be formalized using the CGM in Figure 1.*

Strategies in multi-step games. A *path* in a CGS/CGM is an infinite sequence of states that can result from subsequent transitions in the structure/model. A *strategy* of a player a in a CGS/CGM $\mathcal{M}$ is a conditional plan that specifies what a should do in each possible situation. Depending on the type of memory that we assume for the players, a strategy can be *memoryless*, formally represented with a function $s_a : St \to Act$, such that $s_a(q) \in d_a(q)$, or a *perfect recall strategy*, represented with a function $s_a : St^+ \to Act$ such that $s_a(\langle\dots, q\rangle) \in d_a(q)$, where St^+ is the set of *histories*, i.e., finite prefixes of paths in $\mathcal{M}$ [3, 10]. A *collective strategy* for a group of players $C = \{a_1, ..., a_r\}$ is simply a tuple of strategies $s_C = \langle s_{a_1}, ..., s_{a_r}\rangle$, one for each player from C. We denote player a's component of the collective strategy s_C by $s_C[a]$.

We define the function $out(q, s_C)$ to return the set of all paths $\lambda \in St^\omega$ that can be realised when the players in C follow the strategy s_C from state q onward. Formally, for memoryless strategies, it can be defined as below:

$out(q, s_C) = \{\lambda = q_0, q_1, q_2... \mid q_0 = q$ and for each $i = 0, 1, ...$ there exists $\langle\alpha^i_{a_1}, ..., \alpha^i_{a_k}\rangle$ such that $\alpha^i_a \in d_a(q_i)$ for every $a \in \mathbb{A}\text{gt}$, and $\alpha^i_a \in s_C[a](q_i)$ for $a \in C$, and $q_{i+1} = o(q_i, \alpha^i_{a_1}, ..., \alpha^i_{a_k})\}$.

The definition for perfect recall strategies is analogous.

2.2 Abstract models of coalitional effectivity

Effectivity functions have been introduced in cooperative game theory [7] to provide an abstract representation of the powers of coalitions to influence the outcome of the game.

Definition 3 (Effectivity functions and models). *A* local effectivity function $E : \mathcal{P}(\mathbb{A}\mathrm{gt}) \to \mathcal{P}(\mathcal{P}(St))$ *associates a family of sets of states with each set of players.*

A global effectivity function $E : St \times \mathcal{P}(\mathbb{A}\mathrm{gt}) \to \mathcal{P}(\mathcal{P}(St))$ *assigns a local effectivity function to every state* $q \in St$. *We will use the notations* $E(q)(C)$ *and* $E_q(C)$ *interchangeably.*

Finally, a coalitional effectivity model *consists of a global effectivity function, plus a valuation of atomic propositions.*

Intuitively, elements of $E(C)$ are *choices* available to the coalition C: if $X \in E(C)$ then by choosing X the coalition C can force the outcome of the game to be in X. The idea to represent a choice (action) of a coalition by the set of possible outcomes which can be effected by that choice was also captured by the notion of 'alternating transition system' used originally to provide semantics for ATL in [2].

Definition 4 (True playability [9, 5]). *A local effectivity function E is* truly playable *iff the following hold:*

Outcome monotonicity: *$X \in E(C)$ and $X \subseteq Y$ implies $Y \in E(C)$;*

Liveness: *$\emptyset \notin E(C)$;*

Safety: *$St \in E(C)$;*

Superadditivity: *if $C \cap D = \emptyset$, $X \in E(C)$ and $Y \in E(D)$, then $X \cap Y \in E(C \cup D)$;*

$\mathbb{A}\mathrm{gt}$***-maximality:*** *$\overline{X} \notin E(\emptyset)$ implies $X \in E(\mathbb{A}\mathrm{gt})$;*

Determinacy: *if $X \in E(\mathbb{A}\mathrm{gt})$ then $\{x\} \in E(\mathbb{A}\mathrm{gt})$ for some $x \in X$.*

A global effectivity function is truly playable iff it consists only of local functions that are truly playable.

α-Effectivity. Each strategic game G can be canonically associated with an effectivity function, called the α-effectivity function of G and denoted with E^{α}_{G} [9].

Definition 5 (α-effectivity in strategic games). *For a strategic game G, the (coalitional)* α-effectivity function $E^{\alpha}_{G} : \mathcal{P}(\mathbb{A}\mathrm{gt}) \to \mathcal{P}(\mathcal{P}(St))$ is defined as follows: $X \in E^{\alpha}_{G}(C)$ if and only if there exists σ_C such that for all $\sigma_{\overline{C}}$ we have $o(\sigma_C, \sigma_{\overline{C}}) \in X$.

Example 2. *The α-effectivity for M_1, q_0 is:* $E(\{1,2\}) = \{\{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$; $E(\{1\}) = E(\{2\}) = \{\{q_0, q_1\}, \{q_0, q_2\}, \{q_0, q_1, q_2\}\}$; $E(\emptyset) = \{\{q_0, q_1, q_2\}\}$. *Clearly, E is truly playable.*

Theorem 1 (Representation Theorem [9, 5]). *A local effectivity function E is truly playable if and only if there exists a strategic game G such that $E^{\alpha}_{G} = E$.*

2.3 Logical reasoning about multi-step games

The Alternating-time Temporal Logic ATL* [2, 3] is a multimodal logic with strategic modalities $\langle\!\langle C \rangle\!\rangle$ and temporal operators X ("at the next state"), G ("always from now on"), and $\mathcal{U}$ ("until"). There are two types of formulae of ATL*, *state formulae* and *path formulae*, respectively defined by the following grammar:

$\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\!\langle C \rangle\!\rangle\gamma$,
$\gamma ::= \varphi \mid \neg\gamma \mid \gamma\wedge\gamma \mid X\gamma \mid G\gamma \mid \gamma\,\mathcal{U}\,\gamma$, for $C \subseteq \mathbb{A}\mathrm{gt}$, $\mathsf{p} \in Prop$.
F ("sometime in the future") can be defined as $F\varphi \equiv \top\,\mathcal{U}\,\varphi$.

Let M be a CGM, q a state in M, and $\lambda = q_0, q_1, \ldots$ a path in M. For every $i \in \mathbb{N}$ we denote $\lambda[i] = q_i$; $\lambda[0..i]$ is the prefix $q_0, q_1, \ldots, q_i$, and $\lambda[i..\infty]$ is the respective suffix of λ.

The semantics of ATL* is given by the following clauses [3]:

$M, q \models \mathsf{p}$ iff $q \in V(\mathsf{p})$, for $\mathsf{p} \in Prop$;

$M, q \models \neg\varphi$ iff $M, q \not\models \varphi$;

$M, q \models \varphi_1 \wedge \varphi_2$ iff $M, q \models \varphi_1$ and $M, q \models \varphi_2$;

$M, q \models \langle\!\langle C \rangle\!\rangle\gamma$ iff there is a strategy s_C for the players in C such that for each path $\lambda \in out(q, s_C)$ we have $M, \lambda \models \gamma$.

$M, \lambda \models \varphi$ iff $M, \lambda[0] \models \varphi$;

$M, \lambda \models \neg\gamma$ iff $M, \lambda \not\models \gamma$;

$M, \lambda \models \gamma_1 \wedge \gamma_2$ iff $M, \lambda \models \gamma_1$ and $M, \lambda \models \gamma_2$;

$M, \lambda \models X\gamma$ iff $M, \lambda[1, \infty] \models \gamma$;

$M, \lambda \models G\gamma$ iff $M, \lambda[i, \infty] \models \gamma$ for every $i \geq 0$; and

$M, \lambda \models \gamma_1 \mathcal{U} \gamma_2$ iff there is i such that $M, \lambda[i, \infty] \models \gamma_2$ and $M, \lambda[j, \infty] \models \gamma_1$ for all $0 \leq j < i$.

Example 3. *Consider again the repeated matching pennies from Example 1. No player can make sure that the token moves to any particular position (e.g., $M_1, q_0 \models \neg\langle\!\langle 1 \rangle\!\rangle F\mathsf{pos}_1$). On the other hand, the player can at least make sure that the game will* avoid *particular positions: $M_1, q_0 \models \langle\!\langle 1 \rangle\!\rangle G\neg\mathsf{pos}_1$. And, if the players cooperate then they control the game completely: $M_1, q_0 \models \langle\!\langle 1, 2 \rangle\!\rangle X\mathsf{pos}_0 \wedge \langle\!\langle 1, 2 \rangle\!\rangle X\mathsf{pos}_1 \wedge \langle\!\langle 1, 2 \rangle\!\rangle X\mathsf{pos}_2$.*

ATL and CL as fragments of ATL*. The most important fragment of ATL* is ATL where each strategic modality is directly followed by a single temporal operator. Thus, the semantics of ATL can be given entirely in terms of states, cf. [3] for details. We point out that for ATL the two notions of strategy (memoryless vs. perfect recall) yield the same semantics.

Furthermore, Coalition Logic (CL) [9] can be seen as the fragment of ATL involving only booleans and operators $\langle\!\langle C \rangle\!\rangle X$, and thus it inherits the semantics of ATL on CGMs.

3. STATE EFFECTIVITY IN MULTI-STEP GAMES

An alternative semantics of CL has been given in [9] in terms of the effectivity models defined in section 2.2, via the following clause, where $\varphi^M := \{s \in St \mid M, s \models \varphi\}$.

$M, q \models \langle\!\langle C \rangle\!\rangle X\varphi$ iff $\varphi^M \in E_q(C)$.

It is easy to see that the CGM-based and effectivity-based semantics of CL coincide on truly playable models.

The semantics of ATL has never been explicitly defined in terms of abstract effectivity models. An informal outline of such semantics has been suggested in [4], essentially by representation of the modalities $\langle\!\langle C \rangle\!\rangle G$ and $\langle\!\langle C \rangle\!\rangle \mathcal{U}$ as appropriate fixpoints of $\langle\!\langle C \rangle\!\rangle X$. The idea was based on the result from [3] showing that the alternation-free fragment of Alternating μ-Calculus is strictly more expressive that ATL. In this section, we actually extend state-based effectivity models to provide semantics for ATL. For that, as pointed out earlier, a different effectivity function will be needed for each temporal pattern.

We note that an effectivity function for the "always" modality G was already constructed in [9]. Moreover, an effectivity function for reachability, i.e. for the F modality, has been presented in [1]. Our construction here differs significantly from both approaches, and allows to cover all kinds of effectivity that can be addressed in ATL.

3.1 Operations on state effectivity functions

First, we define basic operations on effectivity functions, reflecting the meaning of these as operations on games.

Composition of effectivity functions $E, F : St \times \mathcal{P}(\mathbb{A}\text{gt}) \to \mathcal{P}(\mathcal{P}(St))$ is the effectivity function $E \circ F$ where $Y \in (E \circ F)_q(C)$ iff there exists a subset Z of St, such that $Z \in E_q(C)$ and $Y \in F_z(C)$ for every $z \in Z$.

Union of effectivity functions E, F is the effectivity function $E \cup F$ where $Y \in (E \cup F)_q(C)$ iff $Y \in E_q(C)$ or $Y \in F_q(C)$. *Intersection* of effectivity functions is defined analogously. Likewise, we define union and intersection of any family of effectivity functions. Hereafter, we assume that $\circ$ has a stronger binding power than $\cup$ and $\cap$.

Inclusion of effectivity functions is defined as follows: $E \subseteq F$ iff $E_q(C) \subseteq F_q(C)$ for every $q \in St$ and $C \subseteq \mathbb{A}\text{gt}$.

The *idle effectivity function* $\mathbf{I}$ is defined as follows: $\mathbf{I}_q(C) = \{Y \subseteq St \mid q \in Y\}$ for every $q \in St$ and $C \subseteq \mathbb{A}\text{gt}$.

PROPOSITION 2. *The following hold for any outcome monotone effectivity functions E, F, G :*

1. $E \circ \mathbf{I} = \mathbf{I} \circ E = E$.
2. *If* $F_1 \subseteq F_2$ *then* $E \circ F_1 \subseteq E \circ F_2$.
3. $(E \cup F) \circ G = (E \circ G) \cup (E \circ F)$.
4. $(E \cap F) \circ G = (E \circ G) \cap (E \circ F)$.

REMARK 3. *The identities* $E \circ (F \cup G) = (E \circ F) \cup (E \circ G)$ *and* $E \circ (F \cap G) = (E \circ F) \cap (E \circ G)$ *are not valid.*

DEFINITION 6. *For any effectivity function E we define inductively the effectivity functions $E^{(n)}$ and $E^{[n]}$ as follows:*
$E^{(0)} = \mathbf{I}$, $E^{(n+1)} = \mathbf{I} \cup E \circ E^{(n)}$,
$E^{[0]} = \mathbf{I}$, $E^{[n+1]} = \mathbf{I} \cap E \circ E^{[n]}$.

PROPOSITION 4. *For every* $n \geq 0 : E^{(n)} \subseteq E^{(n+1)}$ *and* $E^{[n+1]} \subseteq E^{[n]}$.

PROOF. Routine, by induction on n. □

DEFINITION 7. *The* weak iteration *of E is the function* $E^{(*)} = \bigcup_{k=0}^{\infty} E^{(k)}$, *i.e.,* $Y \in E_q^{(*)}(C)$ *iff* $\exists n.\ Y \in E_q^{(n)}(C)$.

The strong iteration *of E is the function* $E^{[*]} = \bigcap_{k=0}^{\infty} E^{[k]}$, *i.e.,* $Y \in E_q^{[*]}(C)$ *iff* $\forall n.\ Y \in E_q^{[n]}(C)$.

PROPOSITION 5. *Unions, intersections, compositions, week and strong iterations preserve outcome-monotonicity of effectivity functions.*

PROPOSITION 6. *For any effectivity function E :*

1. $E^{(*)}$ *is the least fixed point of the monotone operator* $\mathfrak{F}_w$ *defined by* $\mathfrak{F}_w(F) = \mathbf{I} \cup E \circ F$.
2. $E^{[*]}$ *is the greatest fixed point of the monotone operator* $\mathfrak{F}_q$ *defined by* $\mathfrak{F}_q(F) = \mathbf{I} \cap E \circ F$.

PROOF. **(1)** First, we show by induction on k that for every k, $E^{(k)} \subseteq \mathbf{I} \cup E \circ E^{(*)}$. Indeed, $E^{(0)} = \mathbf{I} \subseteq \mathbf{I} \cup E \circ E^{(*)}$; $E^{(k+1)} = \mathbf{I} \cup E \circ E^{(k)} \subseteq \mathbf{I} \cup E \circ E^{(*)}$ by the inductive hypothesis and proposition 2. Thus, $E^{(*)} \subseteq \mathbf{I} \cup E \circ E^{(*)}$.

For the converse inclusion, let $Y \in (\mathbf{I} \cup E \circ E^{(*)})_q(C)$. If $Y \in \mathbf{I}_q(C)$, then $Y \in E_q^{(*)}$ by definition. Suppose $Y \in (E \circ E^{(*)})_q(C)$. Then, there is $Z \in E_q(C)$ such that for every $z \in Z$, $Y \in {E^{(*)}}_z(C)$, hence $Y \in E_z^{(k_z)}(C)$ for some $k_z \geq 0$. Let $m = \max_{z \in Z} k_z$. Then, by proposition 4, $Y \in E_z^{(m)}(C)$ for every $z \in Z$. Therefore, $Y \in (E \circ E^{(m)})_q(C) \subseteq E_q^{(m+1)}(C) \subseteq E_q^{(*)}(C)$.

Thus, $E^{(*)}$ is a fixed point of the operator $\mathfrak{F}_w$.

Now, suppose that F is such that $\mathfrak{F}_w(F) = \mathbf{I} \cup E \circ F$. Then, we show by induction on k that for every k, $E^{(k)} \subseteq F$. Indeed, $E^{(0)} = \mathbf{I} \subseteq \mathbf{I} \cup E \circ F = F$. Suppose $E^{(k)} \subseteq F$. Then $E^{(k+1)} = \mathbf{I} \cup E \circ E^{(k)} \subseteq \mathbf{I} \cup E \circ F = F$ by the inductive hypothesis and proposition 2. Thus, $E^{(*)} \subseteq F$. Therefore, $E^{(*)}$ is the least fixed point of $\mathfrak{F}_w$.

(2). The argument is dually analogous. □

3.2 Binary effectivity functions

DEFINITION 8. *Given a set of players $\mathbb{A}\text{gt}$ and a set of states St, a* local binary effectivity function *for $\mathbb{A}\text{gt}$ on St is a mapping $U : \mathcal{P}(\mathbb{A}\text{gt}) \to \mathcal{P}(\mathcal{P}(St) \times \mathcal{P}(St))$ associating with each set of players a family of pairs of outcome sets.*

A global binary effectivity function *associates a local binary effectivity function with each state from St.*

Now we define some basic (global) binary effectivity functions and operations on them.

Left-idle binary effectivity function $\mathbf{L} : St \times \mathcal{P}(\mathbb{A}\text{gt}) \to \mathcal{P}(\mathcal{P}(St) \times \mathcal{P}(St))$ is defined by $\mathbf{L}_q(C) = \{(X, Y) \mid q \in X\}$ for any $q \in St$ and $C \subseteq \mathbb{A}\text{gt}$. Respectively, *right-idle* binary effectivity function $\mathbf{R}$ is defined by $\mathbf{R}_q(C) = \{(X, Y) \mid q \in Y\}$ for any $q \in St$ and $C \subseteq \mathbb{A}\text{gt}$.

Union of binary effectivity functions $U, W : St \times \mathcal{P}(\mathbb{A}\text{gt}) \to \mathcal{P}(\mathcal{P}(St) \times \mathcal{P}(St))$ is the binary effectivity function $U \cup W$ where $(X, Y) \in (U \cup W)_q(C)$ iff $(X, Y) \in U_q(C)$ or $(X, Y) \in V_q(C)$. *Intersection* of binary effectivity functions is defined analogously. *Right projection* of U is the unary effectivity function E such that $E_q(C) = \{Y \mid (X, Y) \in U_q(C)\}$ for all q, C. Likewise, we define union, intersection, and right projection of any family of binary effectivity functions.

Composition of a unary effectivity function E with a binary effectivity function U is the binary effectivity function $E \circ U$ such that $(X, Y) \in (E \circ U)_q(C)$ iff there exists a subset Z of St, such that $Z \in E_q(C)$ and $(X, Y) \in U_z(C)$ for every $z \in Z$. *Inclusion* of binary effectivity functions: $U \subseteq W$ iff $U_q(C) \subseteq W_q(C)$ for every $q \in St$ and $C \subseteq \mathbb{A}\text{gt}$.

DEFINITION 9. *For any unary effectivity function E we define the binary effectivity functions $E^{\{n\}}$, $n \geq 0$, inductively as follows:* $E^{\{0\}} = \mathbf{R}; E^{\{n+1\}} = R \cup (L \cap E \circ E^{\{n\}})$.

Then, the binary iteration *of E is defined as the binary effectivity function* $E^{\{*\}} = \bigcup_{k=0}^{\infty} E^{\{k\}}$, *i.e.* $(X, Y) \in E_q^{\{*\}}(C)$ *iff* $(X, Y) \in E_q^{\{n\}}(C)$ *for some n.*

DEFINITION 10. *A binary effectivity function U is* outcome-monotone *if every $U_q(C)$ is upwards closed, i e.* $(X, Y) \in U_q(C)$ *and* $X \subseteq X', Y \subseteq Y'$ *imply* $(X', Y') \in U_q(C)$.

Proposition 7. *For any finite set of states St and unary effectivity function E, $E^{\{*\}}$ is the least fixed point of the monotone operator $\mathfrak{F}_b$ defined by $\mathfrak{F}_b(U) = \mathbf{R} \cup (\mathbf{L} \cap E \circ U)$.*

Proof. Analogous to the proof of proposition 6. □

Proposition 8. *$E^{(*)}$, $E^{[*]}$ and $E^{\{*\}}$ are outcome-monotone. Moreover, $E^{(*)}$ is the right projection of $E^{\{*\}}$.*

3.3 State-based effectivity models for ATL

The semantics of ATL can now be given in terms of models that are more abstract and technically simpler than CGM.

Definition 11. *A* state-based effectivity frame (SEF) for ATL *is a tuple $\mathcal{F} = \langle \mathbb{A}\text{gt}, St, \mathbf{E}, \mathbf{G}, \mathbf{U}\rangle$, where: $\mathbb{A}$gt is a set of players, St is a set of states, $\mathbf{E}$ and $\mathbf{G}$ are outcome-monotone effectivity functions, and $\mathbf{U}$ is an outcome-monotone binary effectivity function.*

A state-based effectivity model (SEM) for ATL *is a SEF plus a valuation of atomic propositions.*

Definition 12. *A SEF $\mathcal{F}$ is* standard *iff (1) $\mathbf{E}$ is truly playable, (2) $\mathbf{G} = \mathbf{E}^{[*]}$, and (3) $\mathbf{U} = \mathbf{E}^{\{*\}}$.*

A SEM $\mathcal{M} = \langle \mathcal{F}, V\rangle$ is standard *if $\mathcal{F}$ is standard.*

Now, we define truth of an ATL formula at a state of a state-based effectivity model uniformly as follows:

$\mathcal{M}, q \models \langle\langle C\rangle\rangle X\varphi$ iff $\varphi^{\mathcal{M}} \in \mathbf{E}_q(C)$,

$\mathcal{M}, q \models \langle\langle C\rangle\rangle G\varphi$ iff $\varphi^{\mathcal{M}} \in \mathbf{G}_q(C)$,

$\mathcal{M}, q \models \langle\langle C\rangle\rangle \psi\mathcal{U}\varphi$ iff $(\psi^{\mathcal{M}}, \varphi^{\mathcal{M}}) \in \mathbf{U}_q(C)$.

Extending α-Effectivity to SEM. Given a CGM $M = (\mathbb{A}\text{gt}, St, Act, d, o, V)$, we construct its corresponding SEM as follows: $\text{SEM}(M) = (\mathbb{A}\text{gt}, St, \mathbf{E}, \mathbf{G}, \mathbf{U})$ where $\mathbf{E}_q = E(q)^{\alpha}_{M}$ for all $q \in St$, $\mathbf{G} = \mathbf{E}^{[*]}$ and $\mathbf{U} = \mathbf{E}^{\{*\}}$.

Example 4. *The "always" effectivity in state q_0 of the repeated matching pennies can be written as follows: $\mathbf{G}_{q_0}(\{1,2\}) = \{\{q_0\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_0, q_1, q_2\}\}$, $\mathbf{G}_{q_0}(\{1\}) = \mathbf{G}_{q_0}(\{2\}) = \{\{q_0, q_1\}, \{q_0, q_2\}, \{q_0, q_1, q_2\}\}$, $\mathbf{G}_{q_0}(\emptyset) = \{\{q_0, q_1, q_2\}\}$.*

The next result easily follows from Theorem 1:

Theorem 9 (Representation Theorem). *A state effectivity model $\mathcal{M}$ for ATL is standard iff there exists a CGM M such that $\mathcal{M} = \text{SEM}(M)$.*

Moreover, we observe that the ATL semantics in CGMs and in their associated standard SEMs coincide.

Proposition 10. *For every CGM M, state q in M, and ATL formula φ, we have that $M, q \models \varphi$ iff $\text{SEM}(M), q \models \varphi$.*

Proof. Routine, by structural induction on formulae. □

Corollary 11. *Any ATL formula φ is valid (resp., satisfiable) in concurrent game models iff φ is valid (resp., satisfiable) in standard state-based effectivity models.*

4. COALITIONAL PATH EFFECTIVITY

State-based effectivity models for ATL partly characterize coalitional powers for achieving long-term objectives. However, the applicability of such models is limited by the fact that they characterize effectivity with respect to outcome states, while effectivity for outcome *paths (i.e., plays)* is only captured when such paths are described by the specific temporal patterns definable in ATL. Thus, in particular, state-based effectivity models are not suitable for providing semantics of the whole ATL*.

In this section we aim at getting to the core of the notion of effectivity in multi-step games, regardless of the temporal pattern that defines the winning condition, by re-defining it in terms of outcome *paths*, rather than states. The idea is natural: every collective strategy of the grand coalition in a multi-step game determines a unique path (play) through the state space of the game. Consequently, the outcome of following an individual or coalitional strategy in such game is a set of paths (plays) that can result from execution of the strategy, depending on the moves of the remaining players. Hence, powers of players and coalitions in multi-step games can be characterized by sets of sets of paths. We claim that the notion of path effectivity captures adequately the meaning of strategic operators in ATL(*). Moreover, it provides correct semantics for the whole ATL*, and not only its limited fragment ATL.

4.1 Frames, models, effectivity functions

Definition 13 (Path effectivity function). *Let $\mathbb{A}$gt be a set of players, and St a set of states. A* path *in St is any infinite sequence of states of St. The set of all paths in St is therefore denoted by St^{ω}. A* path effectivity function *is a mapping $\mathcal{E} : \mathcal{P}(\mathbb{A}\text{gt}) \to \mathcal{P}(\mathcal{P}(St^{\omega}))$ that assigns to each coalition a non-empty family of sets of paths.*

The intuition is analogous to that for state effectivity: a set of paths X is in $\mathcal{E}(C)$ means that the coalition C can choose a strategy that ensures that the game will develop along one of the paths in X. Note that this notion refers to global effectivity only: $X \in \mathcal{E}(C)$ can include paths starting from different states. Local effectivity is easily extractable from the global one. This is in line with the concept of a strategy as a complete conditional plan: in particular, the strategy must prescribe collective actions of the coalition from all possible initial states of the game.

Also, we will assume that $\mathcal{E}$ captures the *actual effectivity*, i.e., it collects only the actual outcome paths of choices available to C, and is not necessarily closed under upwards monotonicity. We note that the outcome-monotone notion of effectivity has a somewhat negative meaning, in the sense that $X \in E(C)$ is usually interpreted as "the coalition C can ensure that the outcome of the game *cannot be outside* X", whereas this does not mean that every element of X is a feasible outcome. This distinction is conceptual, rather than technical, but it will influence our construction of effectivity functions for concrete models. The "actuality" assumption is necessary to specify appropriate abstract playability conditions characterizing path effectivity in concrete models.

Definition 14 (Path effectivity frames/models). *A* path effectivity frame (PEF) *is a structure $\mathcal{F} = (\mathbb{A}\text{gt}, St, \mathcal{E})$ consisting of a set of players $\mathbb{A}$gt, a set of states St and a path effectivity function $\mathcal{E}$ on these. A* path effectivity model (PEM) *expands a PEF by a valuation of the propositions $V : Prop \to \mathcal{P}(St)$.*

By analogy with identifying choices as sets of outcome states in state effectivity models, we will refer to sets of paths in a PEF as 'choices', with the intuition that $\mathcal{E}(C)$

defines the strategic choices of the coalition C in a PEF $\mathcal{F}$ as sets of paths in $\mathcal{F}$ that C can enforce. However, not every such path can be a *feasible* outcome in some concrete model (i.e, a CGM), but only those that follow existing transitions in the CGM. So, given a path effectivity frame $\mathcal{F}$, we define the set of 'feasible' paths in $\mathcal{F}$ as

$$Paths_{\mathcal{F}} = \bigcup_{C \subseteq \mathbb{A}\mathrm{gt}} \bigcup_{X \in \mathcal{E}(C)} X.$$

For a PEM $\mathcal{M} = (\mathcal{F}, V)$, we define $Paths_{\mathcal{M}} = Paths_{\mathcal{F}}$.

4.2 Path effectivity in concurrent games

Not every set of paths is a feasible choice for a coalition. Note that the powers of players and coalitions in a game crucially depend on their available strategies. There are different notions of strategy, e.g., depending on the amount of memory they can use, so we will parameterize the new notion of α-effectivity with a type (class) of strategies. Every class of individual strategies of players gives rise to a class of coalitional strategies obtained by freely combining the individual strategies of the participating players.[2] We say that a class Σ of individual and coalitional strategies is *feasible* if every coalition has at least one strategy in Σ. Two types of feasible strategies are especially relevant: *perfect recall* and *memoryless* strategies (introduced in Section 2.1). We will refer to them with $\mathfrak{mem}$ and $\mathfrak{nomem}$, respectively.

Definition 15 (Σ-effectivity). *Let M be a CGM and $\Sigma = \bigcup_{C \subseteq \mathbb{A}\mathrm{gt}} \Sigma_C$ be a feasible set of coalitional strategies in M. The* Σ-effectivity function of M *is defined as*

$$\mathcal{E}^{\Sigma}_M(C) = \{ \bigcup_{q \in St} out(q, s_C) \mid s_C \in \Sigma_C\}.$$

Specifically, we denote by $\mathcal{E}^{\mathfrak{mem}}_M$ and $\mathcal{E}^{\mathfrak{nomem}}_M$ the effectivity of coalitions respectively for perfect recall strategies and for memoryless strategies in M. Note that $\mathcal{E}^{\Sigma}_M$ collects only the *actual* outcome paths of *actual* choices of coalitions in M.

Example 5. *The difference between perfect recall and memoryless effectivity is most easily seen in the case of the grand coalition: $\mathcal{E}^{\mathfrak{mem}}_M = \{\{\lambda\} \mid \lambda \in \{q_0, q_1, q_2\}^\omega\}$, but $\mathcal{E}^{\mathfrak{nomem}}_M = \{\{(q_i)^\omega\} \mid i \in \{0,1,2\}\} \cup \{\{q_i(q_j)^\omega\}\} \cup \{\{(q_iq_j)^\omega\}\} \cup \{\{q_iq_j(q_k)^\omega\} \mid i \neq j\} \cup \{\{q_i(q_jq_k)^\omega\} \mid i \neq j\} \cup \{\{(q_iq_jq_k)^\omega\} \mid i \neq j\}$. That is, the players can enforce any sequence of states when they have perfect memory, but only the "periodic" ones in the memoryless case.*

For a class Σ of strategies in a CGM M we denote the set of paths feasible with respect to Σ in M by $Paths^{\Sigma}_M$. For any path effectivity function $\mathcal{E}$ we denote the set of all feasible paths in $\mathcal{E}$, i.e., appearing in any choice of $\mathcal{E}$, by $Paths_{\mathcal{E}}$.

Proposition 12. *For every CGM M and a feasible class Σ of coalitional strategies in M:*

1. *Every coalition has a collective strategy, and therefore for every state q in M it can enforce at least one, non-empty set of outcome paths starting from q.* (Safety)
2. *For any coalition C in M, every coalitional strategy produces a non-empty set of outcome paths.* (Liveness)
3. *$\mathcal{E}^{\Sigma}_M(\emptyset)$ is a singleton. More precisely, $\mathcal{E}^{\Sigma}_M(\emptyset) = \{Paths^{\Sigma}_M\}$.*
4. *$\mathcal{E}^{\Sigma}_M(\mathbb{A}\mathrm{gt})$ consists of singleton sets. More precisely, $\mathcal{E}^{\Sigma}_M(\mathbb{A}\mathrm{gt}) = \{\{\lambda\} \mid \lambda \in Paths^{\Sigma}_M\}$.* (Determinacy)
5. *Every two disjoint coalitions can join their chosen coalitional strategies to enforce an outcome set of paths that is included in the outcome set of paths enforceable by each of the coalitions following its respective strategy.* (Superadditivity)

[2]Here we adhere to the standard ATL assumption that players in a coalition execute their parts of the joint strategy independently.

To define "playability" conditions for path effectivity frames, we will need some additional notation. Let $q \in St$, $h, h' \in St^+$, $X \in \mathcal{P}(St^\omega)$, and $\mathcal{X} \in \mathcal{P}(\mathcal{P}(St^\omega))$. Then we denote:

- $h \preceq h'$ if h' is an extension of h;
- $X(q) := \{\lambda \in X \mid \lambda[0] = q\}$; $X[i] := \{\lambda[i] \mid \lambda \in X\}$
- $X(h) := \{\lambda \mid \lambda \in X,$ and $\lambda[0..k] = h$ for some $k\}$ is the set of paths in X starting with h;
- $X|h := \{\lambda[k..\infty] \mid \lambda \in X$ and $\lambda[0..k] = h\}$ is the set of suffixes of paths in X, extending h;
- $\mathcal{X}(q) = \{X(q) \mid X \in \mathcal{X}\}$, $\mathcal{X}(h) = \{X(h) \mid X \in \mathcal{X}\}$, and $\mathcal{X}|h = \{X|h \mid X \in \mathcal{X}\}$.

4.3 Path effectivity semantics of ATL*

Given an ATL* path formula γ and a path effectivity model M, let

$$\gamma^M = \{\lambda \in Paths_M \mid M, \lambda \models \gamma\}.$$

denote the set of paths in M that satisfy γ. Note that relation $M, \lambda \models \gamma$ is already well defined by the relevant semantic clauses in Section 2.3 (it is essentially the semantics of Linear Time Logic LTL). Then, the path effectivity semantics of ATL* is given by the clause below:

$M, q \models \langle\!\langle C \rangle\!\rangle \gamma$ iff there is $X \in \mathcal{E}(C)$ such that $X(q) \subseteq \gamma^M$.

Equivalently: $M, q \models \langle\!\langle C \rangle\!\rangle \gamma$ iff $\gamma^M(q) \in \widehat{\mathcal{E}}(C)(q)$, where $\widehat{\mathcal{E}}(C) = \{X \mid Y \subseteq X$ for some $Y \in \mathcal{E}(C)\}$ is the *outcome-monotone closure* of $\mathcal{E}$.

Thus, path effectivity models yield a conceptually simple and technically elegant semantics of ATL*, where one effectivity function suffices to completely describe the powers of coalitions to enforce any ATL*-definable behaviour. In particular, only one simple semantic clause is needed to define strategic ability in ATL*, because the temporal patterns are appropriately handled by LTL semantics.

Example 6. *Like before, we have $\mathcal{E}_{M_1}, q_0 \models \langle\!\langle 1, 2 \rangle\!\rangle F \mathsf{pos_i}$ for every $i = 0, 1, 2$ in both perfect recall and memoryless strategies. This can be demonstrated e.g. by the choice $\{q_0(q_i)^\omega\}$ that belongs to $\mathcal{E}^{\mathfrak{mem}}_{M_1}(\{1,2\})$ as well as $\mathcal{E}^{\mathfrak{nomem}}_{M_1}(\{1,2\})$.*

4.4 Characterizing path effectivity functions

The path effectivity semantics for ATL* defined above is too abstract to be of practical use. Here we identify the characteristic properties of path effectivity functions arising in CGMs and define an analogue of the notion of (truly) playable state effectivity functions.

Definition 16 (Playability in path effectivity). *A path effectivity function $\mathcal{E} : \mathcal{P}(\mathbb{A}\mathrm{gt}) \to \mathcal{P}(\mathcal{P}(St^\omega))$ is* actually playable *if it satisfies the following conditions:*

P-Safety: *$\mathcal{E}(C)(q)$ is non-empty for every $C \subseteq \mathbb{A}\mathrm{gt}, q \in St$.*

P-Liveness: *$\emptyset \notin \mathcal{E}(C)(q)$ for every $C \subseteq \mathbb{A}\mathrm{gt}, q \in St$.*

P-Monotonicity: *For every $C_1 \subseteq C_2 \subseteq \mathbb{A}\mathrm{gt}$ and for every $X_2 \in \mathcal{E}(C_2)$ there is a $X_1 \in \mathcal{E}(C_1)$ such that $X_2 \subseteq X_1$.*

P-Superadditivity: *if $C \cap D = \emptyset$, $X \in \mathcal{E}(C)$ and $Y \in \mathcal{E}(D)$, then $Z \in \mathcal{E}(C \cup D)$ for some $Z \subseteq X \cap Y$.*

P-$\emptyset$-Minimality: *$\mathcal{E}(\emptyset)$ is the singleton $\{Paths_{\mathcal{E}}\}$.*

P-Determinacy: *$\mathcal{E}(\mathbb{A}\mathrm{gt})(q) \subseteq \{\{\lambda\} \mid \lambda \in Paths_{\mathcal{E}}\}$.*[3]

We note that the playability conditions above are variants of true playability, sans outcome monotonicity, for path-oriented effectivity. These conditions can be adapted to state effectivity to provide abstract characterization for actual state effectivity in CGMs, analogous to Pauly's characterization of the outcome-monotone state effectivity in [9]. For lack of space we do not include that result here.

Besides the general conditions in Definition 16, we need additional constraints which are specific to the underlying class of strategies, and relate local choices with global strategies in path effectivity frames.

Path effectivity functions for memoryless strategies

The following definition formalizes the consistency of a family of local choices with a global memoryless strategy.

Definition 17 (nomem-consistent family of choices). *Given a PEF $\mathcal{F} = (\mathbb{A}\mathrm{gt}, St, \mathcal{E})$ and a coalition $C \subseteq \mathbb{A}\mathrm{gt}$, let $\mathcal{X} = \{X^q\}_{q \in St}$ be such that $X^q \in \mathcal{E}(C)(q)$ for every $q \in St$. We call $\mathcal{X}$* nomem-consistent *if for every pair of states $q_1, q_2 \in St$, path $\lambda \in X^{q_1}$, and a prefix $h = \lambda[0..i]$ for some $i \in \mathbb{N}$ such that $\lambda[i] = q_2$, it holds that $X^{q_1}|h = X^{q_2}$.*

Definition 18 (nomem-realizability). *A playable path effectivity function $\mathcal{E} : \mathcal{P}(\mathbb{A}\mathrm{gt}) \to \mathcal{P}(\mathcal{P}(St^\omega))$ is* nomem-realizable *if it also satisfies the following conditions:*

nomem-***Regularity:*** *For every $C \subseteq \mathbb{A}\mathrm{gt}$ and $X \in \mathcal{E}(C)$, the family $\{X(q)\}_{q \in St}$ is nomem-consistent.*

nomem-***Convexity:*** *Let $\{X^q\}_{q \in St}$ s.t. $X^q \in \mathcal{E}(C)(q)$ for all $q \in St$, be nomem-consistent. Then the set of all paths generated by the relation $\{(q, q') \mid q \in St, q' \in X^q[1]\}$ is in $\mathcal{E}(C)$.*

LimitClosure: *Every $X \in \mathcal{E}(C)$ is* limit-closed, *i.e., for every path λ, if every $\lambda[0..i]$, for $i \in \mathbb{N}$, is a prefix of some path $\lambda_i \in X$, then $\lambda \in X$.*

Thus, nomem-regularity of $\mathcal{E}$ means that when coalition C follows a fixed memoryless strategy which determines a set of outcome paths X, it is effective for the same set of outcome paths starting from every occurrence of q along any path from X, viz. $X(q)$. Moreover, nomem-convexity requires that any consistent collection of 'locally applied' strategies for a given coalition C can be pieced together into a global memoryless strategy for C.

Theorem 13 (nomem-Representation theorem). *A path effectivity function $\mathcal{E}$ equals $\mathcal{E}_M^{\mathrm{nomem}}$ for some concurrent game model M if and only if it is nomem-realizable.*

[3]Unlike the case of state effectivity functions, where the determinacy constraint is only needed for infinite state games (cf. [5]), it becomes essential here, because even very simple 2-state structures can generate uncountably many paths.

Proof. (*Sketch*) Showing that for every CGM M, $\mathcal{E}_M^{\mathrm{nomem}}$ is nomem-realizable is fairly routine. For the converse implication, given a nomem-realizable path effectivity function $\mathcal{E}$ we first define a global actual state effectivity function AE by collecting for every C and q the successor states from each set of paths in $\mathcal{E}(C)(q)$ and thus producing $AE(C)(q)$. Then we produce the respective global state effectivity function E by closing AE under upwards monotonicity. It is straightforward to show that E is truly playable by the actual path playability of $\mathcal{E}$. Then using the representation theorem for truly playable state effectivity functions in [5] we construct a CGM M for the same set of agents $\mathbb{A}\mathrm{gt}$ and state space St, such that the α-effectivity function E_M^α of M coincides with E. Using M we construct the respective path effectivity function $\mathcal{E}_M^{\mathrm{nomem}}$ according to Definition 15. Finally, we show that it coincides with $\mathcal{E}$ by using the nomem-realizability of each of $\mathcal{E}$ and $\mathcal{E}_M^{\mathrm{nomem}}$. □

Path effectivity functions for perfect recall strategies

Here we provide a partial characterization of path effectivity functions for perfect recall strategies.

Definition 19 (mem-consistent family of choices). *Given a PEF $\mathcal{F} = (\mathbb{A}\mathrm{gt}, St, \mathcal{E})$ and a coalition $C \subseteq \mathbb{A}\mathrm{gt}$, let $\mathcal{X} = \{X^h\}_{h \in St^+}$ be such that $X^h \in \mathcal{E}(C)(h)$ for every $h \in St^+$. We call $\mathcal{X}$* mem-consistent *if for every pair $h_1, h_2 \in St^+$ such that $h_1 \preceq h_2$, we have $X^{h_1}(h_2) = X^{h_2}$.*

Definition 20 (mem-Realizable effectivity). *A playable path effectivity function $\mathcal{E} : \mathcal{P}(\mathbb{A}\mathrm{gt}) \to \mathcal{P}(\mathcal{P}(St^\omega))$ is* mem-realizable *if it also satisfies the following conditions for every coalition C:*

mem-***Regularity:*** *For every $C \subseteq \mathbb{A}\mathrm{gt}$ and $X \in \mathcal{E}(C)$, the family $\{X(q)\}_{q \in St}$ is mem-consistent.*

mem-***Convexity:*** *Let $\{X^h\}_{h \in St^+}$, where $X^h \in \mathcal{E}(C)(h)$ for every $h \in St^+$, be mem-consistent. Then the set of all paths generated by the relation*
$\{(q, q') \mid q = X^h|h[0], q' \in X^h|h[1]\}$ is in $\mathcal{E}(C)$.

LimitClosure: *as in Definition 18.*

Intuition for the mem-Regularity condition: if C can be efective for $X|h$ after history h, they can obtain it right at the beginning of the game starting from $q = last(h)$. This is because every substrategy of a perfect recall strategy is also a viable perfect recall strategy. Intuition for the mem-Convexity condition: any 'consistent' collection of history-based local choices for a given coalition C can be pieced together into a global perfect recall strategy for C.

The following is easy to check.

Proposition 14. *For every CGM M, $\mathcal{E}_M^{\mathrm{mem}}$ is mem-realizable.*

For lack of space we defer the respective representation theorem for perfect recall strategies to a further work.

5. BEYOND PERFECT INFORMATION

So far, we have only been concerned with games where every player knows the global state of the system at any moment. Modeling and reasoning about imperfect information scenarios is more sophisticated. First, not all strategies are executable – even in the perfect recall case. This is because the agents cannot specify that they will execute two different actions in situations that look the same to them. Therefore, only *uniform* strategies are admissible here. Moreover, it is often important to find a uniform strategy that succeeds in *all* indistinguishable for the agent states, rather than contend that there is such a successful strategy for the current state of the system.

In this section, we briefly sketch how path effectivity models can be used to give account on powers of coalitions under imperfect information. This is by no means intended as an exhaustive analysis. Rather, we point out that the modeling power of path effectivity can be applied to more sophisticated scenarios than ones assuming complete knowledge.

We take Schobbens' ATL$_{ir}$ [10] as the "core", minimal ATL-based language for strategic ability under imperfect information. ATL$_{ir}$ includes the same formulae as ATL, only the cooperation modalities are presented with a subscript: $\langle\!\langle A\rangle\!\rangle_{ir}$ to indicate that they address agents with imperfect **i**nformation and imperfect **r**ecall. Models of ATL$_{ir}$ are *imperfect information concurrent game models* (iCGM), which can be seen as concurrent game models augmented with a family of indistinguishability relations $\sim_a \subseteq St \times St$, one per agent $a \in \mathbb{A}\mathrm{gt}$. The relations describe agents' uncertainty: $q \sim_a q'$ means that, while the system is in state q, agent a considers it possible that it is in q'.

A *uniform strategy* for agent a is a function $s_a : St \to Act$, such that: (1) $s_a(q) \in d(a, q)$; (2) if $q \sim_a q'$ then $s_a(q) = s_a(q')$. A collective strategy is uniform if it contains only uniform individual strategies. Again, function $out(q, s_A)$ returns the set of all paths that may result from agents A executing strategy s_A from state q onward. The semantics of cooperation modalities in ATL$^*_{ir}$ is defined as follows:

$M, q \models \langle\!\langle A\rangle\!\rangle_{ir}\gamma$ iff there exists a uniform collective strategy s_A such that, for each $a \in A$, q' such that $q \sim_a q'$, and path $\lambda \in out(s_A, q')$, we have $M, \lambda \models \gamma$.

First, we observe that the same type of effectivity functions can be used to model powers in imperfect information games: $\mathcal{E} : \mathcal{P}(\mathbb{A}\mathrm{gt}) \to \mathcal{P}(\mathcal{P}(St^\omega))$. Moreover, the notion of Σ-effectivity does not change. Given an iCGM M and $\Sigma = \bigcup_{C\subseteq\mathbb{A}\mathrm{gt}} \Sigma_C$ be a set of (uniform) coalitional strategies in M, the Σ-*effectivity function of* M is still defined as $\mathcal{E}^\Sigma_M(C) = \{\bigcup_{q\in St} out(q, s_C) \mid s_C \in \Sigma_C\}$.

The semantics of ATL$^*_{ir}$ is also very similar to the perfect information case:

$M, q \models \langle\!\langle C\rangle\!\rangle_{ir}\gamma$ iff there is $X \in \mathcal{E}(C)$ such that

$$\bigcup_{a\in C} \bigcup_{q':q\sim_a q'} X(q') \subseteq \gamma^M.$$

That is, $\langle\!\langle C\rangle\!\rangle_{ir}\gamma$ if C have a single choice satisfying γ on all outcome paths starting from states that look the same as q.

What changes is the structural properties of actual effectivity functions that are induced by iCGM's. $\mathbb{A}\mathrm{gt}$-maximality and determinacy are no longer valid since even the grand coalition cannot always enforce every possible course of events, cf. [6]. The regularity and convexity conditions must also be revised because the standard fixpoint characterizations of the temporal modalities do not hold anymore under imperfect information [6]. The detailed study of appropriate realizability conditions is subject to future research.

6. CONCLUSIONS

In this paper we have developed the idea of characterizing multi-player multi-step games in terms of what sets of outcomes – states or paths – coalitions can ensure by executing one or another collective strategy. These characterizations lead to respective notions of state-based and path-based coalition effectivity models, which provide alternative semantics for logics of such games, most notably ATL and ATL*. We find such characterizations both conceptually important and technically interesting because they extract the core game-theoretic essence from game models. They also resolve some technical issues arising in the original semantics for ATL*, particularly in the cases of incomplete and imperfect information. We believe that the understanding of abstract realizability under imperfect information can lead to satisfiability checking procedures and complete axiomatic characterization for these variants of ATL.

Acknowledgements. Wojciech Jamroga acknowledges the support of the FNR (National Research Fund) Luxembourg under project S-GAMES – C08/IS/03.

7. REFERENCES

[1] N. Alechina, B. Logan, N. Nga, and A. Rakib. A logic for coalitions with bounded resources. In *Proceedings of IJCAI*, pages 659–664, 2009.

[2] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. In *Proceedings of FOCS*, pages 100–109, 1997.

[3] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Journal of the ACM*, 49:672–713, 2002.

[4] V. Goranko and W. Jamroga. Comparing semantics of logics for multi-agent systems. *Synthese*, 139(2):241–280, 2004.

[5] V. Goranko, W. Jamroga, and P. Turrini. Strategic games and truly playable effectivity functions. In *Proceedings of AAMAS2011*, pages 727–734, 2011.

[6] W. Jamroga and N. Bulling. Comparing variants of strategic ability. In *Proceedings of IJCAI-11*, pages 252–257, 2011.

[7] H. Moulin and B. Peleg. Cores of effectivity functions and implementation theory. *Journal of Mathematical Economics*, 10(1):115–145, 1982.

[8] M. Osborne and A. Rubinstein. *A Course in Game Theory.* MIT Press, 1994.

[9] M. Pauly. *Logic for Social Software.* PhD thesis, University of Amsterdam, 2001.

[10] P. Y. Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2):82–93, 2004.

Session 5F
Logic and Verification

A logic of emotions: from appraisal to coping

Mehdi Dastani
Utrecht University, The Netherlands
mehdi@cs.uu.nl

Emiliano Lorini
IRIT-CNRS, Toulouse, France
lorini@irit.fr

ABSTRACT

Emotion is a cognitive mechanism that directs an agent's thoughts and attention to what is relevant, important, and significant. Such a mechanism is crucial for the design of resource-bounded agents that must operate in highly-dynamic, semi-predictable environments and which need mechanisms for allocating their computational resources efficiently. The aim of this work is to propose a logical analysis of emotions and their influences on an agent's behavior. We focus on four emotion types (*viz.*, hope, fear, joy, and distress) and provide their logical characterizations in a modal logic framework. As the intensity of emotion is essential for its influence on an agent's behavior, the logic is devised to represent and reason about graded beliefs, graded goals and intentions. The belief strength and the goal strength determine the intensity of emotions. Emotions trigger different types of coping strategy which are aimed at dealing with emotions either by forming or revising an intention to act in the world, or by changing the agent's interpretation of the situation (by changing beliefs or goals).

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Intelligent agents

General Terms

Theory

Keywords

Cognitive models, logic-based approaches and methods

1. INTRODUCTION

Autonomous software agents are assumed to have different (possibly conflicting) objectives, able to sense their environments, update their states accordingly, and decide which actions to perform at any moment in time. The behavior of such software agents can be effective and practical only if they are able to continuously and adequately assess their (sensed) situation and update their states with relevant information and crucial objectives. For example, a robot with a plan to transport a container to its target position may perceive it has low battery charge. The robot may assess the state of its battery charge as being relevant for the objective of having the container at its target position, and update its state by suspending the current battery-demanding transport plan. Such assessment and update may cause the agent to decide to charge its battery right away or to focus on a less battery-demanding task.

Emotion is a (cognitive) mechanism that directs one's thoughts and attention to what is relevant, important, and significant in order to ensure effective behavior. The aim of this work is to propose a logical analysis of the relationships between emotion and cognition. An understanding of these relationships is particularly important in the perspective of the design of resource-bounded agents that must survive in highly-dynamic, semi-predictable environments and which need mechanisms for allocating their computational resources efficiently. Indeed, as it has been stressed by several authors in psychology and economics, emotions provide heuristics for preventing excessive evaluation and deliberation by pruning of search spaces [5] and for interrupting normal cognition when unattended goals require servicing [20], and signals for belief revision [15].

Our approach is inspired by the appraisal and coping models of human emotions [16, 11, 8]. According to these models, an agent continuously appraises its situation (*e.g.*, low battery charge endangers the objective of having a container at its target position) after which emotions can be triggered (*e.g.*, fear of failing to place the container at its target position). The triggered emotions can affect the agent's behavior depending on their intensities. There are often a set of strategies that can be used to cope with a specific emotion, for example, by updating the agent's mental state (*e.g.*, being fearful that the transportation plan will not place the container at its target position leads the agent to reconsider its plan).

We first propose, in Section 2, a dynamic logic with special operators which allow to represent the intentions of a cognitive agent as well as its beliefs and goals with their corresponding strengths. Then, in Section 3, we provide a logical analysis of the intensity of different emotions such as hope, fear, joy and distress. In Section 4, we extend the logic with special operators to formally characterize different kinds of coping strategies which are aimed at dealing with emotions either by forming or revising an intention to act in the world, or by changing the agent's interpretation of the situation (by changing belief strength or goal strength). A complete axiomatization and a decidability result for the logic are given in Section 5. Related works are discussed in Section 6.

2. LOGICAL FRAMEWORK

This section presents the syntax and the semantics of the logic DL-GA (*Dynamic Logic of Graded Attitudes*). This logic is designed to represent beliefs, goals, and intentions, where beliefs and goals have degree of plausibility and desirability, respectively.

2.1 Syntax

Assume a finite set of atomic propositions describing facts $Atm = \{p, q, \ldots\}$, a finite set of physical actions (*i.e.*, actions modifying the physical world) $PAct = \{a, b, \ldots\}$, a finite set of natural numbers

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

$Num = \{x \in \mathbb{N} : 0 \leq x \leq \mathsf{max}\}$, with $\mathsf{max} \in \mathbb{N} \setminus \{0\}$. We note $Num^- = \{-x : x \in Num \setminus \{0\}\}$ the corresponding set of negative integers. We note Lit the set of literals and $l, l', \ldots$ the elements of Lit. Finally, we define the set of propositional formulas $Prop$ to be the set of all Boolean combinations of atomic propositions.

The language $\mathcal{L}$ of DL-GA is defined by the following grammar in Backus-Naur Form (BNF):

$$\begin{array}{lcll} Act & : & \alpha & ::= \; a \mid *\varphi \\ Fml & : & \varphi & ::= \; p \mid \mathsf{exc}_\mathsf{h} \mid \mathsf{Des}^\mathsf{k} l \mid \mathsf{Int}_a \mid \\ & & & \quad\;\; \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{K}\varphi \mid [\alpha]\varphi \end{array}$$

where p ranges over Atm, h ranges over Num, l ranges over Lit, k ranges over $Num \cup Num^-$, and a ranges over $PAct$. The other Boolean constructions $\top$, $\bot$, $\vee$, $\rightarrow$ and $\leftrightarrow$ are defined in the standard way.

The set of actions Act includes both physical actions and sensing actions of the form $*\varphi$. A sensing action is an action which consists in modifying the agent's beliefs in the light of a new incoming evidence. In particular, $*\varphi$ is the mental action (or process) of learning that φ is true. As we will show in Section 2.3, technically this amounts to an operation of belief conditioning in Spohn's sense [21].

The set of formulas Fml contains special constructions exc_h, $\mathsf{Des}^\mathsf{k} l$ and Int_a which are used to represent the agent's mental state. Formulas exc_h are used to identify the degree of *plausibility* of a given world for the agent. We here use the notion of plausibility first introduced by Spohn [21]. Following Spohn's theory, the worlds that are assigned the smallest numbers are the most plausible, according to the beliefs of the individual. That is, the number h assigned to a given world rather captures the degree of *exceptionality* of this world, where the exceptionality degree of a world is nothing but the opposite of its plausibility degree (*i.e.*, the exceptionality degree of a world decreases when its plausibility degree increases). Therefore, formula exc_h can be read alternatively as "the current world has a degree of exceptionality h" or "the current world has a degree of plausibility $\mathsf{max} - \mathsf{h}$".

Formula $\mathsf{Des}^\mathsf{k} l$ has to be read "the state of affairs l has a degree of desirability k for the agent". Degree of desirability can be positive, negative or equal to zero.[1] Suppose $\mathsf{k} > 0$. Then $\mathsf{Des}^\mathsf{k} l$ means that "the agent wishes to achieve l with strength k", whereas $\mathsf{Des}^{-\mathsf{k}} l$ means that "the agent wishes to avoid l with strength k". $\mathsf{Des}^0 l$ means that "the agent is indifferent about l" (*i.e.*, the agent does not care whether l is true or false). For notational convenience, in what follows we will use the following abbreviations:

$$\begin{array}{l} \mathsf{AchG}^\mathsf{k} l \overset{\mathsf{def}}{=} \mathsf{Des}^\mathsf{k} l \text{ for } \mathsf{k} > 0 \\ \mathsf{AvdG}^\mathsf{k} l \overset{\mathsf{def}}{=} \mathsf{Des}^{-\mathsf{k}} l \text{ for } \mathsf{k} > 0 \end{array}$$

where AchG andAvdG respectively stand for *achievement goal* and *avoidance goal*.

Formulas Int_a capture the agent's intentions. We assume that the agent's intentions are only about physical actions and not about sensing actions. The formula Int_a has to be read "the agent has the intention to perform the physical action a" or "the agent is committed to perform the physical action a".

The logic DL-GA has also epistemic operators and modal operators that are used to describe the effects of a given action α. The formula $[\alpha]\varphi$ has to be read "after the occurrence of the action α, φ will be true". $\mathsf{K}\varphi$ has to be read "the agent knows that φ is true". This concept of knowledge is the standard S5-notion, partition-based and fully introspective, that is commonly used in computer science and economics [7]. The operator $\widehat{\mathsf{K}}$ is the dual of K, that is, $\widehat{\mathsf{K}}\varphi \overset{\mathsf{def}}{=} \neg\mathsf{K}\neg\varphi$. As we will show in the Section 2.5, the operator K captures a form of 'absolutely unrevisable belief', that is, a form of belief which is stable under belief revision with any new *evidence*.

[1]However, we assume that exceptionality/plausibility and positive desirability are measured on the same scale Num.

2.2 Physical action description

Similarly to Situation Calculus [18], in our framework physical actions are described in terms of their executability preconditions and of their positive and negative effect preconditions. In particular, we define a function

$$Pre : PAct \longrightarrow Prop$$

to map physical actions to their executability preconditions. Using the notion of executability precondition, we define special dynamic operators for physical actions of the form $\langle\langle a \rangle\rangle$, where $\langle\langle a \rangle\rangle\varphi$ has to be read " the physical action a is executable and, φ will be true afterwards":

$$\langle\langle a \rangle\rangle\varphi \overset{\mathsf{def}}{=} Pre(a) \wedge [a]\varphi$$

Moreover, we introduce two functions

$$\begin{array}{l} \gamma^+ : PAct \times Atm \longrightarrow Prop \\ \gamma^- : PAct \times Atm \longrightarrow Prop \end{array}$$

mapping physical actions and atomic propositions to propositional formulas. The formula $\gamma^+(a, p)$ describes the *positive effect preconditions* of action a with respect to p, whereas $\gamma^-(a, p)$ describes the *negative effect preconditions* of action a with respect to p. The former represent the necessary and sufficient conditions for ensuring that p will be true after the occurrence of the physical action a, while the latter represent the necessary and sufficient conditions for ensuring that p will be false after the occurrence of the physical action a. We make the following *coherence assumption*:

(COH_γ) for every $a \in PAct$ and $p \in Atm$, $\gamma^+(a, p)$ and $\gamma^-(a, p)$ must be logically inconsistent.

COH_γ ensures that actions do not have contradictory effects.

2.3 Models and truth conditions

The semantics of the logic DL-GA is a possible world semantics with special functions for exceptionality, desirability and intentions.

Definition 1 (Model). *DL-GA models are tuples* $M = \langle W, \sim, \kappa_{\mathsf{exc}}, \mathcal{D}, \mathcal{I}, \mathcal{V} \rangle$ *where:*

- W *is a nonempty set of possible worlds or states;*
- $\sim$ *is an equivalence relation between worlds in* W*;*
- $\kappa_{\mathsf{exc}} : W \longrightarrow Num$ *is a total function from the set of possible worlds to the set of natural numbers* Num*;*
- $\mathcal{D} : W \times Lit \longrightarrow Num \cup Num^-$ *is a total function from the set of possible worlds to the set of integers* $Num \cup Num^-$*;*
- $\mathcal{I} : W \longrightarrow 2^{PAct}$ *is a total function called* commitment function*, mapping worlds to sets of physical actions;*
- $\mathcal{V} : W \longrightarrow 2^{Atm}$ *is a valuation function.*

As usual, $p \in \mathcal{V}(w)$ means that proposition p is true at world w. The equivalence relation $\sim$, which is used to interpret the epistemic operator K, can be viewed as a function from W to 2^W. Therefore, we can write $\sim(w) = \{v \in W : w \sim v\}$. The set $\sim(w)$ is the agent's *information state* at world w: the set of worlds that the agent imagines at world w. As $\sim$ is an equivalence relation, if $w \sim v$ then the agent has the same information state at w and v (*i.e.*, the agent imagines the same worlds at w and v).

The function κ_{exc} represents a plausibility grading of the possible worlds and is used to interpret the atomic formulas exc_h.

$\kappa_{\mathsf{exc}}(w) = \mathsf{h}$ means that, according to the agent the world w has a degree of exceptionality h or, alternatively, according to the agent the world w has a degree of plausibility $\mathsf{max} - \mathsf{h}$. (Remember that the degree of plausibility of a world is the opposite of its exceptionality degree). The function κ_{exc} allows to model the notion of belief: among the worlds the agent cannot distinguish from a given world w (*i.e.*, the agent's information state at w), there are worlds that the agent considers more plausible than others. For example, suppose that $\sim(w) = \{w, v, u\}$, $\kappa_{\mathsf{exc}}(w) = 2$, $\kappa_{\mathsf{exc}}(u) = 1$ and $\kappa_{\mathsf{exc}}(v) = 0$. This means that $\{w, v, u\}$ is the set of worlds that the agent imagines at world w. Moreover, according to the agent, the world v is strictly more plausible than the world u and the world u is strictly more plausible than the world w (as $\mathsf{max} - 0 > \mathsf{max} - 1 > \mathsf{max} - 2$).

DL-GA models are supposed to satisfy the following *normality* constraint for the plausibility grading to ensure that the agent's beliefs are consistent:

($Norm_{\kappa_{\mathsf{exc}}}$) for every $w \in W$, there is v such that $w \sim v$ and $\kappa_{\mathsf{exc}}(v) = 0$.

The function $\mathcal{D}$ is used to interpret the atomic formulas $\mathsf{Des}^{\mathsf{k}} l$. Suppose $\mathsf{k} > 0$. Then, $\mathcal{D}(w, l) = \mathsf{k}$ means that, at world w, l has a degree of desirability k; whereas $\mathcal{D}(w, l) = -\mathsf{k}$ means that, at world w, l has a degree of desirability $-\mathsf{k}$ — or equivalently, l has a degree of undesirability k —. $\mathcal{D}(w, l) = 0$ means that the agent is indifferent about l.

The function $\mathcal{I}$ is used to interpret the atomic formulas Int_a. For every world $w \in W$, $\mathcal{I}(w)$ identifies the set of physical actions that the agent intends to perform. $\mathcal{I}(w) = \emptyset$ means that the agent has no intention.

Note that in our dynamic setting an agent may be committed to perform an action even though it believes that this is a *suboptimal* choice, *i.e.*, we do not require agents to have intentions because of their desirable consequences. An agent may have an intention without desiring its consequence because, for example, its beliefs and desires may change due to a sensing action. In our running example, the robot may have the intention to transport a container to a given target position, while it believes that this is a suboptimal choice as it has just learnt that it does not have sufficient battery power to accomplish the task.

Definition 2 (Truth conditions). *Given a* DL-GA *model M, a world w and a formula φ, $M, w \models \varphi$ means that φ is true at world w in M. The rules defining the truth conditions of formulas are:*

- $M, w \models p$ *iff* $p \in \mathcal{V}(w)$
- $M, w \models \mathsf{exc}_{\mathsf{h}}$ *iff* $\kappa_{\mathsf{exc}}(w) = \mathsf{h}$
- $M, w \models \mathsf{Des}^{\mathsf{k}} l$ *iff* $\mathcal{D}(w, l) = \mathsf{k}$
- $M, w \models \mathsf{Int}_a$ *iff* $a \in \mathcal{I}(w)$
- $M, w \models \neg\varphi$ *iff not* $M, w \models \varphi$
- $M, w \models \varphi \wedge \psi$ *iff* $M, w \models \varphi$ *and* $M, w \models \psi$
- $M, w \models \mathsf{K}\varphi$ *iff* $M, v \models \varphi$ *for all v such that* $w \sim v$
- $M, w \models [\alpha]\psi$ *iff* $M^{\alpha}, w \models \psi$

where model M^{α} is defined according to Definitions 3 and 5 below.

Definition 3 (Update via physical action). *Given a* DL-GA *model $M = \langle W, \sim, \kappa_{\mathsf{exc}}, \mathcal{D}, \mathcal{I}, \mathcal{V}\rangle$, The update of M by a is defined as $M^a = \langle W, \sim, \kappa_{\mathsf{exc}}, \mathcal{D}, \mathcal{I}^a, \mathcal{V}^a\rangle$ where for all $w \in W$:*

$$\begin{aligned} \mathcal{I}^a(w) &= \mathcal{I}(w) \setminus \{a\} \\ \mathcal{V}^a(w) &= (\mathcal{V}(w) \cup \{p : M, w \models \gamma^+(a, p)\}) \setminus \\ &\quad \{p : M, w \models \gamma^-(a, p)\} \end{aligned}$$

The performance of a physical action a makes the commitment function $\mathcal{I}$ to remove a from the set of intentions. That is, if an agent intends to perform the physical action a, then after the performance of a the agent does not intend to perform a anymore. Of course, the agent may adopt intention a again by, for example, performing an intention update operation (see Section 4.1). Physical actions modify the physical facts via the positive effect preconditions and the negative effect preconditions, defined in Section 2.2. In particular, if the positive effect preconditions of action a with respect to p holds, then p will be true after the occurrence of a; if the negative effect preconditions of action a with respect to p holds, then p will be false after the occurrence of a.[2]

A sensing action updates the agent's information state by modifying the exceptionality degree of the worlds that the agent can imagine. Before defining such a model update, we follow [21] and lift the exceptionality of a possible world to the exceptionality of a formula viewed as a set of worlds.

Definition 4 (Exceptionality degree of a formula). *Let $||\varphi||_w = \{v \in W : M, v \models \varphi$ and $w \sim v\}$. The exceptionality degree of a formula φ at world w, noted $\kappa^w_{\mathsf{exc}}(\varphi)$, is defined as follows:*

$$\kappa^w_{\mathsf{exc}}(\varphi) = \begin{cases} \min\limits_{v \in ||\varphi||_w} \kappa_{\mathsf{exc}}(v) & \text{if } ||\varphi||_w \neq \emptyset \\ \mathsf{max} & \text{if } ||\varphi||_w = \emptyset \end{cases}$$

As expected, the *plausibility* degree of a formula φ, noted $\kappa^w_{\mathsf{plaus}}(\varphi)$, is defined as $\mathsf{max} - \kappa^w_{\mathsf{exc}}(\varphi)$.

Definition 5 (Update via sensing action). *Given a* DL-GA *model $M = \langle W, \sim, \kappa_{\mathsf{exc}}, \mathcal{D}, \mathcal{I}, \mathcal{V}\rangle$. The update of M by the sensing action $*\varphi$ is defined as $M^{*\varphi} = \langle W, \sim, \kappa^{*\varphi}_{\mathsf{exc}}, \mathcal{D}, \mathcal{I}, \mathcal{V}\rangle$ such that for all w:*

$$\kappa^{*\varphi}_{\mathsf{exc}}(w) = \begin{cases} \kappa_{\mathsf{exc}}(w) - \kappa^w_{\mathsf{exc}}(\varphi) & \text{if } M, w \models \varphi \\ Cut_{\mathsf{B}}(\kappa_{\mathsf{exc}}(w) + \delta) & \text{if } M, w \models \neg\varphi \wedge \widehat{\mathsf{K}}\varphi \\ \kappa_{\mathsf{exc}}(w) & \text{if } M, w \models \mathsf{K}\neg\varphi \end{cases}$$

where $\delta \in Num \setminus \{0\}$ and

$$Cut_{\mathsf{B}}(x) = \begin{cases} x & \text{if } 0 \leq x \leq \mathsf{max} \\ \mathsf{max} & \text{if } x > \mathsf{max} \\ 0 & \text{if } x < 0 \end{cases}$$

The action of sensing that φ is true modifies the agent's beliefs as follows.

1. For every world w in which φ is true, the degree of exceptionality of w decreases from $\kappa_{\mathsf{exc}}(w)$ to $\kappa_{\mathsf{exc}}(w) - \kappa^w_{\mathsf{exc}}(\varphi)$, which is the same thing as saying that, degree of plausibility of w increases from $\mathsf{max} - \kappa_{\mathsf{exc}}(w)$ to $\mathsf{max} - (\kappa_{\mathsf{exc}}(w) - \kappa^w_{\mathsf{exc}}(\varphi))$.
2. For every world w in which φ is false:
   (a) if at w the agent can imagine a world in which φ is true, *i.e.* $\widehat{\mathsf{K}}\varphi$, then the degree of exceptionality of w increases from $\kappa_{\mathsf{exc}}(w)$ to $Cut_{\mathsf{max}}(\kappa_{\mathsf{exc}}(w) + \delta)$, which is the same thing as saying that, the degree of plausibility of w decreases from $\mathsf{max} - \kappa_{\mathsf{exc}}(w)$ to $\mathsf{max} - Cut_{\mathsf{max}}(\kappa_{\mathsf{exc}}(w) + \delta)$;

[2]Note that the order of the set theoretic operations in the definition seems to privilege negative effect preconditions; however, due to the *coherence* assumption COH_{γ} made in Section 2.2 the effects of a physical action will never be inconsistent.

(b) if at w the agent cannot imagine a world in which φ is true, *i.e.* $\mathsf{K}\neg\varphi$, then the degree of exceptionality of w does not change.

The condition 2(b) ensures that the agent's plausibility ordering over worlds does not change, if the agent learns something that he cannot imagine.[3] Cut_{B} is a minor technical device, taken from [3], which ensures that the new plausibility assignment fits into the finite set of natural numbers *Num*. The parameter δ is a *conservativeness* index which captures the agent's disposition (or personality trait) to radically change its beliefs in the light of a new evidence. More precisely, the higher is the index δ, and the higher is the agent's disposition to decrease the plausibility degree of those worlds in which the learnt fact φ is false. (When $\delta = \mathsf{max}$, the agent is minimally conservative). We assume that δ is different from 0 in order to ensure that, after learning that p is true, the agent will believe p for every proposition $p \in Atm$ (see validity (5) in Section 2.5 below).

In the sequel we write $\models_{\mathsf{DL\text{-}GA}} \varphi$ to mean that φ is *valid* in DL-GA (φ is true in all DL-GA models).

2.4 Definition of graded belief

Following [21], we define the concept of belief as a formula which is true in all worlds that are maximally plausible (or minimally exceptional).

Definition 6 (Belief, $\mathsf{B}\varphi$). *In model M at world w the agent believes that φ is true, i.e., $M, w \models \mathsf{B}\varphi$, if and only if, for every v such that $w \sim v$, if $\kappa_{\mathsf{exc}}(v) = 0$ then $M, v \models \varphi$.*

The following concept of graded belief is taken from [10]: the strength of the belief that φ is equal to the exceptionality degree of $\neg\varphi$.

Definition 7 (Graded belief, $\mathsf{B}^{\geq \mathsf{h}}\varphi$). *For all $\mathsf{h} \geq 1$, in model M at world w the agent believes that φ with strength at least h, i.e. $M, w \models \mathsf{B}^{\geq \mathsf{h}}\varphi$, if and only if, $\kappa^{w}_{\mathsf{exc}}(\neg\varphi) \geq \mathsf{h}$.*

An agent has the strong belief that φ if and only if, it believes that φ is true with maximal strength max.

Definition 8 (Strong belief, $\mathsf{SB}\varphi$). *In model M at world w the agent strongly believes that φ (or at w the agent is certain that φ is true), i.e., $M, w \models \mathsf{SB}\varphi$, if and only if $\kappa^{w}_{\mathsf{exc}}(\neg\varphi) = \mathsf{max}$.*

As the following proposition highlights, the concepts of belief, graded belief and strong belief semantically defined in Definitions 6-8 are all syntactically expressible in the logic DL-GA.

Proposition 1. *For every DL-GA model M, world w and $\mathsf{h} \in Num$ such that $\mathsf{h} \geq 1$:*

1. $M, w \models \mathsf{B}\varphi$ *iff* $M, w \models \mathsf{K}(\mathsf{exc}_0 \rightarrow \varphi)$
2. $M, w \models \mathsf{B}^{\geq \mathsf{h}}\varphi$ *iff* $M, w \models \mathsf{K}(\mathsf{exc}_{\leq \mathsf{h}-1} \rightarrow \varphi)$
3. $M, w \models \mathsf{SB}\varphi$ *iff* $M, w \models \mathsf{K}(\mathsf{exc}_{\leq \mathsf{max}-1} \rightarrow \varphi)$

where $\mathsf{exc}_{\leq \mathsf{k}} \overset{\mathrm{def}}{=} \bigvee_{0 \leq \mathsf{l} \leq \mathsf{k}} \mathsf{exc}_{\mathsf{l}}$ *for all* $\mathsf{k} \in Num$.

We define the dual operators in the usual way: $\widehat{\mathsf{B}}\varphi \overset{\mathrm{def}}{=} \neg\mathsf{B}\neg\varphi$, $\widehat{\mathsf{B}}^{\geq \mathsf{h}}\varphi \overset{\mathrm{def}}{=} \neg\mathsf{B}^{\geq \mathsf{h}}\neg\varphi$ and $\widehat{\mathsf{SB}}\varphi \overset{\mathrm{def}}{=} \neg\mathsf{SB}\neg\varphi$.

We assume that "the agent believes that φ *exactly* with strength h", *i.e.* $\mathsf{B}^{\mathsf{h}}\varphi$, if and only if the agent believes that φ with strength at least h and it is not the case that the agent believes that φ with strength at least $\mathsf{h}+1$. That is, we define:

$$\mathsf{B}^{\mathsf{h}}\varphi \overset{\mathrm{def}}{=} \mathsf{B}^{\geq \mathsf{h}}\varphi \wedge \neg\mathsf{B}^{\geq \mathsf{h}+1}\varphi \text{ if } 1 \leq \mathsf{h} < \mathsf{max}, \text{ and}$$
$$\mathsf{B}^{\mathsf{max}}\varphi \overset{\mathrm{def}}{=} \mathsf{B}^{\geq \mathsf{max}}\varphi$$

2.5 Some properties of epistemic attitudes

The following validities highlight some interesting properties of beliefs. For every $\mathsf{h}, \mathsf{k} \in Num$ such that $\mathsf{h} \geq 1$ and $\mathsf{k} \geq 1$ we have:

$$\models_{\mathsf{DL\text{-}GA}} \mathsf{K}\varphi \rightarrow \mathsf{B}^{\geq \mathsf{h}}\varphi \tag{1}$$
$$\models_{\mathsf{DL\text{-}GA}} \mathsf{B}\varphi \leftrightarrow \mathsf{B}^{\geq 1}\varphi \tag{2}$$
$$\models_{\mathsf{DL\text{-}GA}} \mathsf{SB}\varphi \leftrightarrow \mathsf{B}^{\geq \mathsf{max}}\varphi \tag{3}$$
$$\models_{\mathsf{DL\text{-}GA}} \neg(\mathsf{B}\varphi \wedge \mathsf{B}\neg\varphi) \tag{4}$$
$$\models_{\mathsf{DL\text{-}GA}} \widehat{\mathsf{K}}\varphi \rightarrow [*\varphi]\mathsf{B}\varphi \text{ if } \varphi \in Prop \tag{5}$$
$$\models_{\mathsf{DL\text{-}GA}} (\mathsf{B}^{\geq \mathsf{h}}\varphi \wedge \mathsf{B}^{\geq \mathsf{k}}\psi) \rightarrow \mathsf{B}^{\geq \min\{\mathsf{h},\mathsf{k}\}}(\varphi \wedge \psi) \tag{6}$$
$$\models_{\mathsf{DL\text{-}GA}} (\mathsf{B}^{\geq \mathsf{h}}\varphi \wedge \mathsf{B}^{\geq \mathsf{k}}\psi) \rightarrow \mathsf{B}^{\geq \max\{\mathsf{h},\mathsf{k}\}}(\varphi \vee \psi) \tag{7}$$

According to the validity (1), knowing that φ implies believing that φ with strength at least h. According to the validity (2), belief is graded belief with strength at least 1. According to the validity (3), the agent has the strong belief that φ if and only if, it believes that φ with maximal strength max. According to the validity (4) (which follows from the normality constraint $NORM_{\kappa_{\mathsf{exc}}}$ in Section 2.3), an agent cannot have inconsistent beliefs. The validity (5) highlights a basic property of belief revision in the sense of AGM theory [2]: if φ is an objective fact and the agent can imagine a world in which φ is true then, after learning that φ is true, the agent believes that φ.[4] According to the validities (6) and (7), if the agent believes that φ with strength at least h and believes that ψ with strength at least k, then the strength of the belief that $\varphi \wedge \psi$ is at least $\min\{\mathsf{h}, \mathsf{k}\}$; if the agent believes that φ with strength at least h and believes that ψ with strength at least k, then it believes $\varphi \vee \psi$ with strength at least $\max\{\mathsf{h}, \mathsf{k}\}$. Similar properties for graded belief are given in possibility theory [6].

3. EMOTIONS AND THEIR INTENSITY

We use the modal operators of graded belief and graded goal of the logic DL-GA to provide a logical analysis of emotions such as hope, fear, joy and distress with their intensities.

According to some psychological models [17, 11, 16] and computational models [9, 4] of emotions, the intensity of hope with respect to a given event is a monotonically increasing function of the degree to which the event is desirable and the likelihood of the event. That is, the higher is the desirability of the event, and the higher is the intensity of the agent's hope that this event will occur; the higher is the likelihood of the event, and the higher is the intensity of the agent's hope that this event will occur. Analogously, the intensity of fear with respect to a given event is a monotonically increasing function of the degree to which the event is undesirable and the likelihood of the event. There are several possible merging functions which satisfy these properties. For example, we could define the merging function *merge* as an average function, according to which the intensity of hope about a certain event is the average of the strength of the belief that the event will occur and the strength of the goal that it will occur. That is, for every $\mathsf{h}, \mathsf{k} \in Num$ representing respectively the strength of the belief and the strength of

[3]Note that the tree conditions 1, 2(a) and 2(b) cover all cases. Indeed, the third condition $\mathsf{K}\neg\varphi$ is equivalent to $\neg\varphi \wedge \mathsf{K}\neg\varphi$, because $\mathsf{K}\neg\varphi \rightarrow \neg\varphi$ is valid.

[4]The only difference with AGM theory is the condition $\widehat{\mathsf{K}}\varphi$. AGM assumes that new information φ must incorporated in the belief base (the so-called success postulate), whereas we here assume that φ must incorporated in the belief base *only if* the agent can imagine a world in which φ is true.

the goal, we could define *merge*(h,k) as $\frac{h+k}{2}$. Another possibility is to define *merge* as a product function (also used in [9, 17]), according to which the intensity of hope about a certain event is the product of the strength of the belief that the event will occur and the strength of the goal that it will occur. Here we do not choose a specific merging function, as this would much depend on the domain of application in which the formal model has to be used.

The emotion intensity scale is defined by the following set:

$$EmoInt = \{y : \text{ there are } x_1, x_2 \in Num \text{ such that } merge(x_1,x_2) = y\}$$

As *Num* is finite, *EmoInt* is finite too.

Let us define the notions of hope and fear with their corresponding intensities. We say that the agent hopes with intensity i that its current intention to perform the action *a* will lead to the desirable consequence *l* if and only if, there are $\mathsf{h}, \mathsf{k} \in Num \setminus \{0\}$ such that $merge(\mathsf{h},\mathsf{k}) = \mathsf{i}$ and $\mathsf{h} < \mathsf{max}$ and: (1) the agent believes with strength h that the physical action *a* is executable and *l* will be true afterwards, (2) the agent wishes to achieve *l* with strength k, (3) the agent intends to perform the physical action *a*. Formally:

$$\mathsf{Hope}^{\mathsf{i}}(a,l) \overset{\text{def}}{=} \bigvee_{\mathsf{h},\mathsf{k}\in Num\setminus\{0\}:\mathsf{h}<\mathsf{max} \text{ and } merge(\mathsf{h},\mathsf{k})=\mathsf{i}} (\mathsf{B}^{\mathsf{h}}\langle\langle a\rangle\rangle l \wedge \mathsf{AchG}^{\mathsf{k}} l \wedge \mathsf{Int}_a)$$

We say that the agent fears with intensity i that its current intention to perform the action *a* will lead to the undesirable consequence *l* if and only if, there are $\mathsf{h}, \mathsf{k} \in Num \setminus \{0\}$ such that $merge(\mathsf{h},\mathsf{k}) = \mathsf{i}$ and $\mathsf{h} < \mathsf{max}$ and: (1) the agent believes with strength h that the action *a* is executable and *l* will be true afterwards, (2) the agent wishes to avoid *l* with strength k, (3) the agent intends to perform the action *a*. Formally:

$$\mathsf{Fear}^{\mathsf{i}}(a,l) \overset{\text{def}}{=} \bigvee_{\mathsf{h},\mathsf{k}\in Num\setminus\{0\}:\mathsf{h}<\mathsf{max} \text{ and } merge(\mathsf{h},\mathsf{k})=\mathsf{i}} (\mathsf{B}^{\mathsf{h}}\langle\langle a\rangle\rangle l \wedge \mathsf{AvdG}^{\mathsf{k}} l \wedge \mathsf{Int}_a)$$

In the preceding definitions of hope and fear, the strength of the belief is supposed to be less than max in order to distinguish hope and fear, which imply some form of uncertainty, from happiness and distress which are based on certainty. Indeed, we have that:

$$\models_{\mathsf{DL\text{-}GA}} \mathsf{Hope}^{\mathsf{i}}(a,l) \rightarrow \neg\mathsf{SB}\langle\langle a\rangle\rangle l \quad (8)$$

$$\models_{\mathsf{DL\text{-}GA}} \mathsf{Fear}^{\mathsf{i}}(a,l) \rightarrow \neg\mathsf{SB}\langle\langle a\rangle\rangle l \quad (9)$$

This means that if an agent hopes/fears that its intention to perform the action *a* will lead to the desirable/undesirable result *l*, then it is not certain about that. For example, if our robot hopes to place a container at a given target position by its transport plan, then the robot is not certain that the container will be at the target position after performing the transport plan. On the contrary, to be joyful/distressed that its current intention to perform the action *a* will lead to the desirable/undesirable consequence *l*, the agent should be *certain* that its intention to perform the action *a* will lead to the desirable/undesirable consequence *l*. This is consistent with OCC psychological model of emotions [16] according to which, while joy and distress are triggered by *actual consequences*, hope and fear are triggered by *prospective consequences* (or *prospects*). Like [9], we here interpret the term 'prospect' as synonymous of 'uncertain consequence' (in contrast with 'actual consequence' as synonymous of 'certain consequence'). The following are our definitions of joy and distress about actions:

$$\mathsf{Joy}^{\mathsf{i}}(a,l) \overset{\text{def}}{=} \bigvee_{\mathsf{k}\in Num\setminus\{0\}:merge(\mathsf{max},\mathsf{k})=\mathsf{i}} (\mathsf{SB}\langle\langle a\rangle\rangle l \wedge \mathsf{AchG}^{\mathsf{k}} l \wedge \mathsf{Int}_a)$$

$$\mathsf{Distress}^{\mathsf{i}}(a,l) \overset{\text{def}}{=} \bigvee_{\mathsf{k}\in Num\setminus\{0\}:merge(\mathsf{max},\mathsf{k})=\mathsf{i}} (\mathsf{SB}\langle\langle a\rangle\rangle l \wedge \mathsf{AvdG}^{\mathsf{k}} l \wedge \mathsf{Int}_a)$$

where $\mathsf{Joy}^{\mathsf{i}}(a,l)$ and $\mathsf{Distress}^{\mathsf{i}}(a,l)$ respectively mean that "the agent is joyful that its current intention to perform the action *a* will lead to the desirable consequence *l*" and "the agent is distressed that its current intention to perform the action *a* will lead to the undesirable consequence *l*". Note that, when computing the intensity of joy and distress, the belief parameter in the merging function *merge* is set to max because strong belief is equivalent to graded belief with maximal strength (validity (3) in Section 2.5).

We here distinguish distress from sadness by adding a condition to the definition of distress: the appraisal variable called *controllability* or *control potential* [19]. That is, to be sad that its current intention to perform the action *a* will lead to the undesirable result *l*, the agent should be certain that it has no control over the undesirable result *l*, in the sense that the agent cannot prevent *l* to be true — which is the same thing as saying that *l* will be true after every executable action of the agent —.

$$\mathsf{Sadness}^{\mathsf{i}}(a,l) \overset{\text{def}}{=} \mathsf{Distress}^{\mathsf{i}}(a,l) \wedge \mathsf{SB}\neg\mathsf{Control}\ l$$

with

$$\mathsf{Control}\ \varphi \overset{\text{def}}{=} \bigvee_{b\in PAct} \langle\langle b\rangle\rangle\neg\varphi$$

where Control φ means "the agent has control over φ" (or "the agent can prevent φ to be true"). Our definition is consistent with some psychological theories [19, 11] according to which, undesirable states of affairs that not be controlled makes one to be sad.

EXAMPLE 1. *Consider again our robot which can decide to transport either container number 1 or container number 2 to a given target position. The former task is more demanding than the latter task, as container number 1 is much heavier than container number 2. In particular, the former task requires at least a full battery charge, whereas the latter requires at least a half battery charge. This means that the action of transporting container number 1 ($transport_1$) and the action of transporting container number 2 ($transport_2$) have the following positive effect preconditions with respect to the objective of placing a container at the target position (pos):*
$\gamma^+(transport_1, pos) = fullCharge,$
$\gamma^+(transport_2, pos) = fullCharge \vee halfCharge.$
Let us assume that the two actions are always executable: $Pre(tranport_1) = Pre(tranport_2) = \top$.

Suppose that at the state w the robot intends to transport container number 1 to the target position and considers undesirable with degree k *not to have any container at the target position,* i.e.,

$$M, w \models \mathsf{AvdG}^{\mathsf{k}}\neg pos \wedge \mathsf{Int}_{transport_1}$$

Moreover, suppose that the robot is certain that in the current situation there is no container at the target position, i.e.,

$$M, w \models \mathsf{SB}\neg pos$$

Finally, suppose that the robot is minimally conservative in revising its beliefs, that is, $\delta = \mathsf{max}$.

The robot observes its battery load and realizes that it does not have a full battery charge but only a half battery charge. After the observation the robot will strongly believe that, if it follows its intention, it will not place any container at the target position, i.e.,

$$M^{*halfCharge\wedge\neg fullCharge}, w \models \mathsf{SB}\langle\langle transport_1\rangle\rangle\neg pos$$

It should be noted that the new model $M^{*halfCharge\wedge\neg fullCharge}$ *is exactly the same as M except for the plausibility value* κ_{exc}. *This implies that we have,*

$$M^{*halfCharge\wedge\neg fullCharge}, w \models \mathsf{AvdG}^{\mathsf{k}}\neg pos \wedge \mathsf{Int}_{transport_1} \wedge \mathsf{SB}\langle\langle transport_1\rangle\rangle\neg pos$$

and therefore for $merge(\mathsf{max},\mathsf{k}) = \mathsf{i}$ *we have*

$$M^{*halfCharge\wedge\neg fullCharge}, w \models \mathsf{Distress}^{\mathsf{i}}(transport_1, \neg pos)$$

This means that, after having observed that it only has a half battery charge, the robot is distressed with intensity i *that, if it follows its current intention, then it will not succeed in placing a container at the target position.*

4. FROM APPRAISAL TO COPING

In the previous section, we have characterized emotions in terms of beliefs, (achievement and avoidance) goals, and intentions, and formalized their intensities in terms of belief strength and goal strength. Emotions with high intensity influence the agent's behavior in order to cope with relevant and significant events. In general, coping can be seen as a cognitive mechanism whose aim is to discharge a certain emotion by modifying one or more of the mental attitudes (*e.g.*, beliefs, goals, intentions) that triggered the emotion [11]. For example, our robot can cope with its distress that if it follows its current intention then it will not succeed in placing a container at the target position, by reconsidering its current intention. The coping mechanism determines various types of responses, also called coping strategies. We here consider three types of coping strategies: coping strategies affecting intentions, coping strategies affecting beliefs and coping strategies affecting goals. More precisely, we consider coping strategies which deal with emotion either by forming or revising an intention to act in the world, or by changing the agent's interpretation of the situation (by changing belief strength or goal strength).

4.1 Coping strategies: syntax and semantics

We extend the logic DL-GA with three different kinds of coping strategies: (1) coping strategies affecting beliefs of the form $\varphi\!\uparrow^{\mathsf{B}}$ and $\varphi\!\downarrow^{\mathsf{B}}$, (2) coping strategies affecting goals of the form $l\!\uparrow^{\mathsf{D}}$ and $l\!\downarrow^{\mathsf{D}}$, and (3) coping strategies affecting intentions of the form $-a$ and $+a$. We call DL-GA$^+$ the resulting logic. $\varphi\!\uparrow^{\mathsf{B}}$ consists in increasing the strength of the belief that φ is true, while $\varphi\!\downarrow^{\mathsf{B}}$ consists in reducing the strength of the belief that φ is true. $l\!\uparrow^{\mathsf{D}}$ consists in increasing the desirability of l, while $l\!\downarrow^{\mathsf{D}}$ consists in reducing the desirability of l. Finally, $-a$ consists in removing the intention Int_a, while $+a$ consists in generating the intention Int_a.

The set of coping strategies is defined by the following grammar:

$$CStr \quad : \quad \beta \quad ::= \quad \varphi\!\uparrow^{\mathsf{B}} \mid \varphi\!\downarrow^{\mathsf{B}} \mid l\!\uparrow^{\mathsf{D}} \mid l\!\downarrow^{\mathsf{D}} \mid -a \mid +a$$

where φ ranges over *Fml*, l ranges over *Lit*, and a ranges over *PAct*. For every coping strategy β we introduce a corresponding dynamic operator $[\beta]$, where $[\beta]\psi$ has to be read "after the occurrence of β, ψ will be true".

As expected, the truth conditions of the new operators are given in terms of model transformation. For every $\beta \in CStr$ we define:

$$M, w \models [\beta]\psi \text{ iff } M^{\beta}, w \models \psi$$

The model M^{β} is defined according to the Definitions 9-11 below.

Coping strategies affecting the strength of the belief that φ either increase or decrease the exceptionality of the worlds in which φ is false with ω unit, only if the agent believes that φ, *i.e.* $\mathsf{B}\varphi$. If the agent does not believe that φ, *i.e.* $\neg\mathsf{B}\varphi$, they do not have any effect on the agent's mental state. In fact, we assume that coping strategies can only operate on *existing* beliefs of the agent by either increasing or decreasing their strengths. ω is a parameter which captures the agent's disposition (or personality trait) to radically change its mental state when coping with emotions (the higher is ω, and the higher is the agent's disposition to change its mental state when coping with emotions).

Definition 9 (Update via coping strategy on beliefs). *Given a DL-GA model M and $\beta \in \{\varphi\!\uparrow^{\mathsf{B}}, \varphi\!\downarrow^{\mathsf{B}}\}$, the updated model M^{β} is defined as $M^{\beta} = \langle W, \sim, \kappa^{\beta}_{\mathsf{exc}}, \mathcal{D}, \mathcal{I}, \mathcal{V}\rangle$ where for all w:*

$$\kappa^{\beta}_{\mathsf{exc}}(w) \quad = \quad \begin{cases} \kappa_{\mathsf{exc}}(w) & \text{if } M, w \models \varphi \\ Cut_{\mathsf{B}}(\kappa_{\mathsf{exc}}(w) + \omega) & \text{if } M, w \models \neg\varphi \wedge \mathsf{B}\varphi \text{ and } \beta = \varphi\!\uparrow^{\mathsf{B}} \\ \kappa_{\mathsf{exc}}(w) & \text{if } M, w \models \neg\varphi \wedge \neg\mathsf{B}\varphi \text{ and } \beta = \varphi\!\uparrow^{\mathsf{B}} \\ Cut_{\mathsf{B}}(\kappa_{\mathsf{exc}}(w) - \omega) & \text{if } M, w \models \neg\varphi \wedge \mathsf{B}\varphi \text{ and } \beta = \varphi\!\downarrow^{\mathsf{B}} \\ \kappa_{\mathsf{exc}}(w) & \text{if } M, w \models \neg\varphi \wedge \neg\mathsf{B}\varphi \text{ and } \beta = \varphi\!\downarrow^{\mathsf{B}} \end{cases}$$

$\omega \in Num \setminus \{0\}$ and Cut_{B} has been defined in Definition 5.

Coping strategies affecting desirability of l either increase or decrease the desirability of l with ω unit.

Definition 10 (Update via coping strategy on goals). *Given a DL-GA model M and $\beta \in \{l\!\uparrow^{\mathsf{D}}, l\!\downarrow^{\mathsf{D}}\}$, the updated model M^{β} is defined as $M^{\beta} = \langle W, \sim, \kappa_{\mathsf{exc}}, \mathcal{D}^{\beta}, \mathcal{I}, \mathcal{V}\rangle$ where for all w:*

$$\mathcal{D}^{\beta}(w, l') \quad = \quad \begin{cases} Cut_{\mathsf{D}}(\mathcal{D}(w, l') + \omega) & \text{if } \beta = l\!\uparrow^{\mathsf{D}} \text{ and } l' = l \\ Cut_{\mathsf{D}}(\mathcal{D}(w, l') - \omega) & \text{if } \beta = l\!\downarrow^{\mathsf{D}} \text{ and } l' = l \\ \mathcal{D}(w, l') & \text{if } l' \neq l \end{cases}$$

$\omega \in Num \setminus \{0\}$ and:

$$Cut_{\mathsf{D}}(y) \quad = \quad \begin{cases} y & \text{if } -\mathsf{max} \leq y \leq \mathsf{max} \\ \mathsf{max} & \text{if } y > \mathsf{max} \\ -\mathsf{max} & \text{if } y < -\mathsf{max} \end{cases}$$

Cut_{D} ensures that the new desirability degree of a literal fits into the finite set of integers $Num \cup Num^{-}$.

Finally, coping strategies affecting intentions change the commitment function by either adding or removing an intention.

Definition 11 (Update via coping strategy on intentions). *Given a DL-GA model M and $\beta \in \{-a, +a\}$, the updated model M^{β} is defined as $M^{\beta} = \langle W, \sim, \kappa_{\mathsf{exc}}, \mathcal{D}, \mathcal{I}^{\beta}, C, \mathcal{V}\rangle$ where for all w:*

$$\mathcal{I}^{\beta}(w) \quad = \quad \begin{cases} \mathcal{I}(w) \setminus \{a\} & \text{if } \beta = -a \\ \mathcal{I}(w) \cup \{a\} & \text{if } \beta = +a \end{cases}$$

The following validities capture some expected properties of coping strategies affecting beliefs and goals. If $\mathsf{h} \geq 1$ then:

$$\models_{\text{DL-GA}^+} \mathsf{B}^{\geq \mathsf{h}}\varphi \rightarrow [\varphi\!\uparrow^{\mathsf{B}}]\mathsf{B}^{\geq Cut_{\mathsf{B}}(\mathsf{h}+\omega)}\varphi \qquad (10)$$

$$\models_{\text{DL-GA}^+} \mathsf{B}^{\geq \mathsf{h}}\varphi \rightarrow [\varphi\!\downarrow^{\mathsf{B}}]\mathsf{B}^{\geq Cut_{\mathsf{B}}(\mathsf{h}-\omega)}\varphi \text{ if } Cut_{\mathsf{B}}(\mathsf{h}-\omega) > 0 \qquad (11)$$

$$\models_{\text{DL-GA}^+} \mathsf{B}^{\geq \mathsf{h}}\varphi \rightarrow [\varphi\!\downarrow^{\mathsf{B}}]\neg\mathsf{B}\varphi \text{ if } Cut_{\mathsf{B}}(\mathsf{h}-\omega) = 0 \qquad (12)$$

$$\models_{\text{DL-GA}^+} \mathsf{Des}^{\mathsf{h}}l \rightarrow [\varphi\!\uparrow^{\mathsf{D}}]\mathsf{Des}^{Cut_{\mathsf{D}}(\mathsf{h}+\omega)}l \qquad (13)$$

$$\models_{\text{DL-GA}^+} \mathsf{Des}^{\mathsf{h}}l \rightarrow [\varphi\!\downarrow^{\mathsf{D}}]\mathsf{Des}^{Cut_{\mathsf{D}}(\mathsf{h}-\omega)}l \qquad (14)$$

4.2 Triggering conditions of coping strategies

In our model coping strategies have triggering conditions which are captured by the function

$$Trg : CStr \longrightarrow Fml$$

mapping coping strategies to DL-GA-formulas. For every coping strategy β, $Trg(\beta)$ captures the conditions under which the coping strategy β is *possibly* triggered. Following current psychological and computational models of emotions [11, 16, 9], we here assume that coping strategies are triggered by the agent's positively valenced emotions (*e.g.*, hope and joy) and negatively valenced emotions (*e.g.*, fear and sadness). In what follows we only discuss coping strategies triggered by negatively valenced emotions.

We assume that an agent that is fearful or distressed because its intention a will realize the undesirable effect l will possibly reconsider its intention. Such an intention reconsideration strategy can be formulated as follows:

$$Trg(-a) = \bigvee_{l \in Lit, \mathsf{i} \in EmoInt: \mathsf{i} \geq \theta}((\mathsf{Fear}^{\mathsf{i}}(a, l) \vee \mathsf{Distress}^{\mathsf{i}}(a, l)) \wedge \mathsf{B}\ \mathsf{Control}\ l)$$

This means that the coping strategy of reconsidering the intention to perform the action a is triggered if and only if (1) the agent is either fearful or distressed with intensity at least θ that its intention to perform the action a will lead to an undesirable result, (2) the agent believes that he has control over l, in the sense that he can prevent the undesirable result l to be true by performing a different action. θ is a threshold which captures the agent's sensitivity to negative emotions (the lower is θ, and the higher is the agent's disposition to discharge a negative emotion by coping with it). Control l captures the appraisal variable called *controllability* we have discussed in Section 3.

Furthermore, we assume that an agent that is fearful or distressed because it believes that its intended action a will realize the undesirable consequence l on which it has no control (1) will decrease the strength of the belief that action a will lead to the undesirable consequence l or, (2) will increase the desirability of l. The former kind of coping strategy captures *wishful thinking* while the latter captures *mental disengagement*.

$$Trg(\langle\langle a\rangle\rangle l{\downarrow}^{\mathsf{B}}) = \bigvee_{l \in Lit, \mathsf{i} \in EmoInt: \mathsf{i} \geq \theta}((\mathsf{Fear}^{\mathsf{i}}(a,l) \vee \mathsf{Distress}^{\mathsf{i}}(a,l)) \wedge \neg\mathsf{B}\ \mathsf{Control}\ l)$$

$$Trg(l{\uparrow}^{\mathsf{D}}) = \bigvee_{l \in Lit, \mathsf{i} \in EmoInt: \mathsf{i} \geq \theta}((\mathsf{Fear}^{\mathsf{i}}(a,l) \vee \mathsf{Distress}^{\mathsf{i}}(a,l)) \wedge \neg\mathsf{B}\ \mathsf{Control}\ l)$$

Note that differently from intention-related coping, wishful thinking and mental disengagement are triggered if the agent appraises that it has no controllability of the undesirable consequence l, in the sense that it cannot prevent l to be true (on this see also [14]).

EXAMPLE 2. *Let us continue the example of Section 3. We have,*

$$M^{*halfCharge \wedge \neg fullCharge}, w \models \mathsf{Distress}^{\mathsf{i}}(transport_1, \neg pos)$$

Suppose $\mathsf{i} \geq \theta$. *Given the assumption that* $Pre(tranport_2) = \top$ *and the positive effect preconditions of* $transport_2$ *with respect to pos, the robot believes that the action* $transport_2$ *will place the second container at the target position,* i.e.,

$$M^{*halfCharge \wedge \neg fullCharge}, w \models \mathsf{B}\langle\langle transport_2\rangle\rangle pos$$

and therefore,

$$M^{*halfCharge \wedge \neg fullCharge}, w \models \mathsf{B}\ \mathsf{Control}\ \neg pos$$

Following the specification of the triggering condition for intention-related coping, the robot can now reconsider its intention $\mathsf{Int}_{transport_1}$,

$$M^{*halfCharge \wedge \neg fullCharge}, w \models Trg(-transport_1)$$

5. AXIOMATIZATION AND DECIDABILITY

The logic DL-GA of Section 2 is axiomatized as an extension of the normal modal logic **S5** for the epistemic operator K with (1) a theory describing the constraints imposed on the agent's mental state, (2) the reduction axioms of the dynamic operators $[\alpha]$, and (3) an inference rule of replacement of equivalents.

Theory of the agent's mental state.

$$\bigvee_{\mathsf{h} \in Num} \mathsf{exc_h}$$
$$\bigvee_{\mathsf{k} \in Num \cup Num^-} \mathsf{Des}^{\mathsf{k}} l$$
$$\mathsf{exc_h} \rightarrow \neg\mathsf{exc_l} \text{ if } \mathsf{h} \neq \mathsf{l}$$
$$\mathsf{Des}^{\mathsf{k}} l \rightarrow \neg\mathsf{Des}^{\mathsf{m}} l \text{ if } \mathsf{k} \neq \mathsf{m}$$
$$\widehat{\mathsf{K}}\ \mathsf{exc}_0$$

Reduction axioms for the dynamic operators $[\alpha]$.

$$[\alpha]p \leftrightarrow \begin{cases} (\gamma^+(a,p) \wedge \neg\gamma^-(a,p)) \vee (p \wedge \neg\gamma^-(a,p)) & \text{if } \alpha = a \\ p & \text{if } \alpha = *\varphi \end{cases}$$

$$[\alpha]\mathsf{Int}_a \leftrightarrow \begin{cases} \bot & \text{if } \alpha = a \\ \mathsf{Int}_a & \text{if } \alpha \neq a \end{cases}$$

$$[\alpha]\mathsf{exc_h} \leftrightarrow \begin{cases} \mathsf{exc_h} \quad \text{if } \alpha = a \\ ((\varphi \wedge \bigvee_{\mathsf{l,m} \in Num \setminus \{0\}: \mathsf{l-m=h}}(\mathsf{B^m}\neg\varphi \wedge \mathsf{exc_l})) \vee \\ (\varphi \wedge (\widehat{\mathsf{B}}\varphi \wedge \mathsf{exc_h})) \vee \\ (\neg\varphi \wedge \widehat{\mathsf{K}}\varphi \wedge \bigvee_{\mathsf{l} \in Num: Cut_{\mathsf{B}}(\mathsf{l}+\delta)=\mathsf{h}} \mathsf{exc_l}) \vee \\ (\mathsf{K}\neg\varphi \wedge \mathsf{exc_h})) \text{ if } \alpha = *\varphi \end{cases}$$

$$[\alpha]\mathsf{Des}^{\mathsf{k}} l \leftrightarrow \mathsf{Des}^{\mathsf{k}} l$$
$$[\alpha]\neg\psi \leftrightarrow \neg[\alpha]\psi$$
$$[\alpha](\psi_1 \wedge \psi_2) \leftrightarrow ([\alpha]\psi_1 \wedge [\alpha]\psi_2)$$
$$[\alpha]\mathsf{K}\psi \leftrightarrow \mathsf{K}[\alpha]\psi$$

Rule of replacement of equivalents.

$$\text{From } \psi_1 \leftrightarrow \psi_2 \text{ infer } \varphi \leftrightarrow \varphi[\psi_1/\psi_2]$$

Given a formula φ, let $red(\varphi)$ be the formula obtained by iterating the application of the reduction axioms from the left to the right, starting from one of the innermost dynamic operators $[\alpha]$. *red* pushes the dynamic operators inside the formula, and finally eliminates them when facing an atomic proposition. Obviously, $red(\varphi)$ does not contain dynamic operators $[\alpha]$. The following proposition is proved using the reduction axioms above and the rule of replacement of equivalents.

PROPOSITION 2. *Let* φ *be a formula in the language of* DL-GA. *Then,* $red(\varphi) \leftrightarrow \varphi$ *is* DL-GA *valid.*

THEOREM 1. *Satisfiability in* DL-GA *is decidable.*

SKETCH OF PROOF. Let L-GA be the fragment of the logic DL-GA without dynamic operators. The problem of satisfiability in L-GA is reducible to the problem of *global* logical consequence in **S5**, where the set of global axioms Γ is the theory of the agent's mental state given above. That is, we have $\models_{\mathsf{L\text{-}GA}} \varphi$ if and only if $\Gamma \models_{\mathbf{S5}} \varphi$. Observe that Γ is finite. It is well-known that the problem of global logical consequence in **S5** with a finite number of global axioms is reducible to the problem of satisfiability in **S5**. The problem of satisfiability checking in **S5** is decidable [7]. It follows that the problem of satisfiability checking in the logic L-GA is decidable too. Proposition 2 and the fact that L-GA is a conservative extension of DL-GA ensure that *red* provides an effective procedure for reducing a DL-GA formula φ into an equivalent L-GA formula $red(\varphi)$. As L-GA is decidable, DL-GA is decidable too.

■

The logic DL-GA$^+$ of Section 4 is axiomatized by the axioms and the rules of inference of the logic DL-GA *plus* the following reduction axioms for the dynamic operators $[\beta]$.

Reduction axioms for the dynamic operators $[\beta]$.

$$[\beta]p \leftrightarrow p$$

$$[\beta]\mathsf{Int}_a \leftrightarrow \begin{cases} \top & \text{if } \beta = +a \\ \bot & \text{if } \beta = -a \\ \mathsf{Int}_a & \text{if } \beta \neq +a \text{ and } \beta \neq -a \end{cases}$$

$$[\beta]\mathsf{exc_h} \leftrightarrow \begin{cases} ((\varphi \vee (\neg\varphi \wedge \neg\mathsf{B}\varphi)) \wedge \mathsf{exc_h}) \vee \\ (\neg\varphi \wedge \mathsf{B}\varphi \wedge \bigvee_{\mathsf{l} \in Num: Cut_{\mathsf{B}}(\mathsf{l}+\omega)=\mathsf{h}} \mathsf{exc_l}) \quad \text{if } \beta = \varphi{\uparrow}^{\mathsf{B}} \\ ((\varphi \vee (\neg\varphi \wedge \neg\mathsf{B}\varphi)) \wedge \mathsf{exc_h}) \vee \\ (\neg\varphi \wedge \mathsf{B}\varphi \wedge \bigvee_{\mathsf{l} \in Num: Cut_{\mathsf{B}}(\mathsf{l}-\omega)=\mathsf{h}} \mathsf{exc_l}) \quad \text{if } \beta = \varphi{\downarrow}^{\mathsf{B}} \\ \mathsf{exc_h} \quad \text{if } \beta = l{\uparrow}^{\mathsf{D}} \text{ or } \beta = l{\downarrow}^{\mathsf{D}} \end{cases}$$

$$[\beta]\mathsf{Des}^{\mathsf{k}} l \leftrightarrow \begin{cases} \bigvee_{\mathsf{m} \in Num \cup Num^-: Cut_{\mathsf{D}}(\mathsf{m}+\omega)=\mathsf{k}} \mathsf{Des}^{\mathsf{m}} l & \text{if } \beta = l{\uparrow}^{\mathsf{D}} \\ \bigvee_{\mathsf{m} \in Num \cup Num^-: Cut_{\mathsf{D}}(\mathsf{m}-\omega)=\mathsf{k}} \mathsf{Des}^{\mathsf{m}} l & \text{if } \beta = l{\downarrow}^{\mathsf{D}} \\ \mathsf{Des}^{\mathsf{k}} l & \text{if } \beta \neq l{\uparrow}^{\mathsf{D}} \text{ and } \beta \neq l{\downarrow}^{\mathsf{D}} \end{cases}$$

$$[\beta]\neg\psi \leftrightarrow \neg[\beta]\psi$$
$$[\beta](\psi_1 \wedge \psi_2) \leftrightarrow ([\beta]\psi_1 \wedge [\beta]\psi_2)$$
$$[\beta]\mathsf{K}\psi \leftrightarrow \mathsf{K}[\beta]\psi$$

The following Theorem 2 is proved in the same way as Theorem 1.

THEOREM 2. *Satisfiability in* DL-GA$^+$ *is decidable.*

6. RELATED WORK

Although psychological models of emotion emphasize the role of emotion intensity and its role in the coping mechanism, most existing works on logical modeling of emotions have ignored either the intensity of emotions or the coping strategies.

Adam et al. [1] have proposed a logical formalization of the OCC model, while Lorini & Schwarzentruber [13] have formalized counterfactual emotions such as regret and disappointment. Both approaches ignore the quantitative aspect of emotions. In a previous work [12] we formalized emotion intensity by using a similar logic, but we did not consider the coping strategies.

The logical approach to emotion proposed by Steunebrink et al. [22] has both characteristics of our approach: it provides a formal model of emotions extended with their intensities and coping strategies. In this model, an intensity function is assigned to each appraised emotion to determine its intensity at each state of the model. The coping mechanism introduced in this model is inspired by Frijda's theory of action tendencies [8]. According to this theory, specific emotions give agents the tendency to perform particular actions. In the proposed model, coping strategies are developed for negative emotions and their aim is to reduce the intensity of negative emotions. However, unlike the present approach, Steunebrink et al.'s approach takes emotion intensity as a primitive without explaining how it depends on more primitive cognitive ingredients such as belief strength and goal strength. The other important difference between the present work and Steunebrink et al.'s work is that we here provide a decidable logic of emotion with a complete axiomatization, whereas Steunebrink et al. do not provide any decidability result or complete axiomatization for their logic of emotion.

In the computational model proposed by Gratch and Marsella [9, 14], the eliciting conditions of emotions are defined in terms of quantitative measures such as desirability and likelihood of events. The model is based on several thresholds that determine when emotions are elicited and how emotions are coped with. The implementation of the proposed model is called EMA and is applied to generate predictions about human emotions and their coping strategies. Since the model is quantitative and the authors do no provide any details about its underlying logic, it is hard to compare this model with other logical approaches. One can only conclude that the model proposed by Gratch and Marsella considers both emotion intensities and coping strategies, although it does not provide a logical characterization of the emotions, their intensities, or the corresponding coping strategies.

7. DISCUSSION

In this work we have provided a logical characterization of emotions enriched with intensities and coping strategies. Emotions are defined in terms of graded beliefs, graded (achievement and avoidance) goals, and intentions. The intensity of emotions, which is defined as a function of belief strength and goal strength, is used to trigger specific coping strategies. We have considered only a few coping strategies triggered by negative emotions. In future work, we intend to extend our analysis to coping strategies triggered by positive emotions. For example, hope or joy with respect to a current intention may trigger coping strategies that suspend the other intentions in order to create a focus on the intended action for which the agent is hopeful or joyful. Moreover, the emotions discussed in this paper are defined with respect to an agent's action. We would like to extend our model in order to characterize emotions in terms of events that are independent from the agent's actions and intentions. Finally, in the present work we have only modeled the so-called prospective emotions, rather than actual emotions. We believe that our model can be easily extended to characterize actual emotions such as being joyful to have already placed a container at the target position or being hopeful that the current state of the battery charge is not empty.

8. REFERENCES

[1] C. Adam, A. Herzig, and D. Longin. A logical formalization of the OCC theory of emotions. *Synthese*, 168(2):201–248, 2009.

[2] C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: partial meet contraction and revision functions. *J. of Symbolic Logic*, 50(2):510–530, 1985.

[3] G. Aucher. A combined system for update logic and belief revision. In *Proc. of PRIMA 2004*, LNAI, pages 1–18. Springer-Verlag, 2005.

[4] T. Bosse and E. Zwanenburg. There's always hope: Enhancing agent believability through expectation-based emotions. In *Proc. of ACII 2009*, pages 111–118. IEEE Computer Society Press, 2009.

[5] A. R. Damasio. *Descartes' Error: Emotion, Reason, and the Human Brain*. Putnam Pub Group, 1994.

[6] D. Dubois and H. Prade. *Possibility theory: an approach to computerized processing of uncertainty*. Plenum Press, 1988.

[7] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[8] N. Frijda. *The Emotions*. Cambridge University Press, 1987.

[9] J. Gratch and S. Marsella. A domain independent framework for modeling emotion. *Cognitive Systems Research*, 5(4):269–306, 2004.

[10] N. Laverny and J. Lang. From knowledge-based programs to graded belief-based programs, part II: off-line reasoning. In *Proc. of IJCAI'05*, pages 497–502, 2005.

[11] R. S. Lazarus. *Emotion and adaptation*. Oxford University Press, 1991.

[12] E. Lorini. A dynamic logic of knowledge, graded beliefs and graded goals and its application to emotion modelling. In *Proc. of LORI-III*, LNCS, pages 165–178. Springer-Verlag, 2011.

[13] E. Lorini and F. Schwarzentruber. A logic for reasoning about counterfactual emotions. *Artificial Intelligence*, 175(3-4):814–847, 2011.

[14] S. Marsella and J. Gratch. EMA: A process model of appraisal dynamics. *Cognitive Systems Research*, 10:70–90, 2009.

[15] W. U. Meyer, R. Reisenzein, and A. Schützwohl. Towards a process analysis of emotions: The case of surprise. *Motivation and Emotion*, 21:251–274, 1997.

[16] A. Ortony, G. L. Clore, and A. Collins. *The cognitive structure of emotions*. Cambridge University Press, 1988.

[17] R. Reisenzein. Emotions as metarepresentational states of mind: naturalizing the belief-desire theory of emotion. *Cognitive Systems Research*, 10:6–20, 2009.

[18] R. Reiter. *Knowledge in action: logical foundations for specifying and implementing dynamical systems*. MIT Press, 2001.

[19] I. J. Roseman, A. A. Antoniou, and P. E. Jose. Appraisal determinants of emotions: constructing a more accurate and comprehensive theory. *Cognition and Emotion*, 10:241–277, 1996.

[20] H. A. Simon. Motivational and emotional controls of cognition. *Psychological Review*, 74:29–39, 1967.

[21] W. Spohn. Ordinal conditional functions: a dynamic theory of epistemic states. In *Causation in decision, belief change and statistics*, pages 105–134. Kluwer, 1998.

[22] B. R. Steunebrink, M. Dastani, and J.-J. Ch. Meyer. A formal model of emotion-based action tendency for intelligent agents. In *Proc. of EPIA'09*, LNAI, pages 174–186. Springer-Verlag, 2009.

Automatic Verification of Epistemic Specifications under Convergent Equational Theories

Ioana Boureanu
Ecole Polytechnique Federal de Lausanne
ioana.boureanu@epfl.ch

Andrew V. Jones
Department of Computing
Imperial College London
andrew.jones@ic.ac.uk

Alessio Lomuscio
Department of Computing
Imperial College London
a.lomuscio@ic.ac.uk

ABSTRACT

We present a methodology for the automatic verification of multi-agent systems against temporal-epistemic specifications derived from higher-level languages defined over convergent equational theories. We introduce a modality called *rewriting knowledge* that operates on local equalities. We discuss the conditions under which its interpretation can be approximated by a second modality that we introduce called *empirical knowledge*. Empirical knowledge is computationally attractive from a verification perspective. We report on an implementation of a technique to verify this modality with the open source model checker MCMAS. We evaluate the approach by verifying multi-agent models of electronic voting protocols automatically extracted from high-level descriptions.

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Verification

General Terms

Security, Verification

Keywords

Epistemic Logic, Equational Rewriting, Model Checking

1. INTRODUCTION

Over the past decade there has been increased interest in developing methodologies for the verification of multi-agent systems (MAS). An approach that has been shown effective is that of symbolic model checking [15, 18] for MAS specified in semantics for temporal-epistemic logic [12]. This has been effectively used in a number of practical applications, including autonomous underwater vehicles [11] and cryptographic protocols [23].

A clear advantage of MAS-based approaches using temporal-epistemic logic is the intuitiveness of the resulting specifications to be checked. Concepts emerging from the MAS community are now being exported to other close disciplines that increasingly see the benefit of using powerful, expressive languages.

One of these areas is *security*. It has long been recognised [5] that cryptographic protocols can benefit from specifications in which knowledge-based concepts feature prominently. Concepts such as anonymity, privacy and non-repudiation can be both naturally and powerfully expressed in epistemic languages. Influential works in the area include the formulation of secrecy by Halpern *et al.* [14], advances on algorithmic knowledge [20] and the epistemic modelling of unlinkability [23]. These have found applications in MAS in a variety of ways, including attack detectability [4].

Still, fundamental problems remain. Firstly, the indistinguishability relations to be used when interpreting the knowledge modalities need to account for the cryptographic primitives used in the messages exchanged. For instance, the set of indistinguishable states should be computed by taking into account the agent's ability to decipher a given message. While some approaches (e.g., [7]) support cryptographic primitives such as encryption and decryption, existing approaches fall short of addressing the more general classes including digital signatures and bit-commitments. Yet, these primitives are prominent in several classes of protocols, e.g., e-voting or zero-knowledge.

Secondly, little or no support for cryptography-inspired modalities is currently provided in existing tools. An extension to MCMAS [15] that caters for explicit knowledge exists [16], but we are unaware of any model checker supporting epistemic modalities for cryptographic concepts, or, indeed, other application-driven epistemic modality of use in many MAS settings. In fact, recent approaches [4] have been restricted to protocols in which receivers can decode all messages down to their atomic constituents immediately upon their receipt. This assumption is not natural in many settings including e-voting, where principals are often only able to decipher messages only at the end of a run.

In this paper we develop an approach aimed at overcoming these limitations. Specifically, in Section 2 we define a novel epistemic modality that is interpreted with respect to *a general equational theory* defining the system. This differs from the standard approach in which the agents' knowledge is interpreted on the equality of the local states. The high computational cost of deducing equivalence under equational theories has been previously discussed [8]. Thus, in Section 3, we put forward a computationally efficient approximation. Section 4 discusses the implementation of this revised modality on top of the model checker MCMAS. In Section 5 we evaluate the techniques presented by verifying e-voting protocols modelled as MAS as per the formalism developed in Section 2. We discuss the results and conclude in Section 6.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

2. A TEMPORAL-EPISTEMIC LOGIC FOR SECURITY PROTOCOLS

In this section we introduce a MAS-based semantics for an epistemic logic under convergent equational theories.

2.1 Preliminaries

We assume familiarity with the concepts presented in this subsection; the following is intended to fix the notation only.

Interpreted Systems. The interpreted systems (IS) formalism [19] is a MAS-based semantics for temporal-epistemic logic (CTLK) [12]. We assume a set $Ag = \{1, \ldots, n\}$ of agents and an *Environment* denoted by *Env*. An agent i is described by a set L_i of possible *local states*, a set Act_i of *local actions*, a *local protocol* function $P_i : L_i \to 2^{Act_i}$ and a *local evolution* function $t_i : L_i \times Act_1 \times \ldots \times Act_n \times Act_{Env} \to L_i$. An IS is defined by the set $G \subseteq \prod_{1 \leq i \leq n} L_i \times L_E$ of *global states*, the set $Act = Act_1 \times \ldots \times Act_n \times Act_{Env}$ of *joint actions*, the *joint protocol* $\overline{P} = (P_1, \ldots, P_n, P_{Env})$ and the *global evolution* function $\overline{t} = (t_1, \ldots, t_n, t_{Env})$. For a global state $g \in G$ and a joint action $a \in Act$, we use g_i and a_i to denote the local state and the local action of agent i, respectively. For more details on interpreted systems, we refer the reader to [12].

Equational Theories [10]. For ease of reference, consider the following equational theory aimed at checking whether one integer is smaller or equal to another.

An Equational Theory (Σ_1, E_1) for Illustrative Purposes.

Signature Σ_1: Sorts: $S = \{nat, bool\}$; Variables: $X_{nat} = \{x, y\}$; Function Symbols: $\Sigma_{\lambda,bool} = \{true, false\}$; $\Sigma_{[(nat,nat),bool]} = \{\leq\}$; $\Sigma_{[bool,bool]} = \{\neg\}$ $\Sigma_{[\lambda,nat]} = \{0\}$; $\Sigma_{[nat,nat]} = \{succ\}$ $\Sigma_{[\omega,s]} = \emptyset$, otherwise (i.e., for other $\omega \in S^*, s \in S$);
The set E_1 of Σ_1-Equations : $((\neg true) = false)$; $((\neg false) = true)$; $(\leq(0, x) = true)$; $(\leq(succ(x), 0) = false)$; $(\leq(succ(x), succ(y)) = \leq(x, y))$;

Let S be a non-empty set of *sorts* (i.e., simple types such as *nat* and *bool* in the example above). Let $\Sigma = \{\Sigma_{(\omega,s)} | \omega \in S^*, s \in S\}$ be an *S-sorted signature*, i.e., a collection of *functional symbols* of *type* $[\omega, s]$. Generally, $\sigma \in \Sigma_{(\omega,s)}$ denotes a *function symbol* (e.g., $\neg, \leq, succ$ in the example above). Let X be an S-sorted set of variables (e.g., x and y above), where X_s are the variables of sort s (e.g., X_{nat} is still $\{x, y\}$ above). Let $T_{\Sigma,X}$ be the S-sorted set of terms over X and Σ (e.g., $succ(x)$ is a term over signature Σ_1 in the example above), and T_Σ be the set of *ground* terms, i.e., terms without variables. An *equational theory* is a tuple (Σ, E), where Σ is a signature and E is a set of Σ-equations. The notation $t = t'$ or $t \to_E t'$ denotes a Σ-*equation*, i.e., a pair (t, t') of terms equal under E.

The semantics for equational theories can be given through the *S-sorted* Σ-*algebra* $\mathbb{A} = (A, \Sigma^A)$ [10], where the set $A = (A_s \,|\, s \in S)$ is an indexed set of values (i.e., the sort *nat* above is mapped under $\mathbb{A}$ onto the set A_{nat} of concrete values, e.g., natural numbers). For each sort s, the set A_s is called the *support-set* for s. The set Σ^A is a set of functions f^A from A_ω to A_s corresponding to function symbols f in Σ, $f \in \Sigma_{(\omega,s)}$, $\omega \in S^*$, $s \in S$ (e.g., the symbol *succ* corresponds to a concrete successor function operating on natural numbers). The indexed set $\delta = (\delta_s \mid s \in S)$ of maps is an *assignment* of X into the algebra $\mathbb{A}$; the tuple $\delta = [x_1/_{v_1}, \ldots, x_n/_{v_n}]$ represents an assignment where the variable x_i is set to the value v_i, for $i \in \{1, \ldots, n\}$. Moreover, the notation $\delta[x/_v]$ represents the assignment obtained from δ when $\delta_s(x)$ is replaced by the value v, for some $x \in X_s$, $v \in A_s$, $s \in S$.

The relation $\to^*_E$ is the transitive closure of $\to_E \cup =$. Let $t \in T_{\Sigma,X}$. The *normal term* of t (denoted by $t{\downarrow_E}$) is the unique term $t' \in T_{\Sigma,Y}$ such that $t \to^*_E t'$, where $Y \subseteq X$. Finally, recall that an equational theory E is *convergent* if the algebra of its semantics can be mechanised into a rewriting system $\to_E$ which is convergent (i.e., *confluent* and *terminating* [2]). A theory is *subterm convergent* if all the subterm in the left-hand side of the rewriting rule also appear on the right-hand side of it. For more details on equational theories, we refer to [10].

Protocols. Security protocols often rely on primitives such as encryption and hashing to establish some security property, e.g., authentication. These primitives can be formally described by equational theories. Consider the simple protocol below constructed on the theory (Σ_1, E_1).

A Communication Protocol Pr_1.

1. $A \!\to B : n$
2. $B \!\to A : m$
3. $A \!\to B : \;\leq(n, m)$

The protocol Pr_1 describes a set of send-receive rules of the two *roles*: the *A-role* and the *B-role*. An agent assuming the *A-role* initiates the protocol by sending the term n to its *B*-role partner agent. This receiver replies with the term m. The initiator terminates the protocol with the acknowledgement $\leq(n, m)$, where $\leq$ is a publicly known function symbol. This symbol is described by the equational theory (Σ_1, E_1) aforementioned. The protocol Pr_1 is purposely simple to exemplify the material that follows.

High-level security languages such as Common Authentication Protocol Specification Language (CAPSL) [9] provide precise descriptions of security protocols, including the underlying equational theory formalising the effects of the protocol-primitives executed by each role (i.e., in our presentation, by some agent assuming that role). We assume that all protocols referred to henceforth are specified in CAPSL.

2.2 Protocol Model

In this subsection we put forward a technique for producing a fully instantiated interpreted system that models a finite number of protocol sessions running concurrently. The aim of this construction is to obtain interpreted systems in which the epistemic relation for each agent is an equivalence relation under the underlying equational theory of the protocol.

Consider a generic security protocol *Pr* that is specified by an equational theory (Σ, E). Its execution generates an instantiation of the protocol. To model this, let a Σ-algebra $\mathbb{A}$, together with a finite set Δ of assignments of variables X in $\mathbb{A}$, be the interpretation of the protocol's theory (Σ, E). Importantly, assume that the rewriting sequence $t \to_{E^*} t{\downarrow_E}$ mechanising the term algebra $\mathbb{T}_{\Sigma,X}$ is somehow provided for each $t \in T_{\Sigma,X}$, e.g., by using a rewriting engine or a CAPSL compiler [9] a priori. Since the normal terms are at hand, to obtain the equivalence between states we will not need to express the message-deducibility relation beforehand, as required in other approaches [8]. Also, we only consider a bounded number of protocol instantiations. By doing so we obtain decidable state-equivalence modulo convergent equational theories.

On the algebra $\mathbb{A}$, let the set $T_{\Sigma,X}|_{Pr}$ denote the terms used only in the actual description of the protocol *Pr*. When *Pr* is implicit, we simply write simply $T_{\Sigma,X}$ to mean $T_{\Sigma,X}|_{Pr}$. In doing so, we underline the protocol-terms (i.e., messages and their subparts) and, later, their values. Thus, for the protocol Pr_1, the set $T_{\Sigma_1,X}|_{Pr_1}$ of terms is $\{A, B, n, m, \leq(n,m)\}$. To highlight variables describing a role of the protocol (i.e., variables A and B in Pr_1), we introduce an additional sort called *role*. Variables of sort *role* (i.e., X_{role}) range over support set A_{role}. This is taken to be a set of strings, for example $\{alice, bob, greg, \ldots\}$.

Protocol Roles to Agents.

An entire protocol-role (e.g., a sender) is described by a variable of sort *role* together with all the terms and actions that inherently characterise it.

Assume a particular assignment $\delta \in \Delta$. An assignment δ (homomorphically) instantiates all the terms in $T_{\Sigma,X}|_{Pr}$. As such, an assignment symbolically corresponds to a *protocol session* (e.g., a session of Pr_1 is given by $\delta = [n/_3, m/_2, A/_{alice}, B_{bob}]$). However, to model the development of the protocol execution more clearly, we will use the projection of each such assignment $\delta \in \Delta$ per each role. Note that when the sender A starts the protocol Pr_1, it does not possess any value for the variable m or for the term $\leq(n,m)$ (as it will only receive m after the protocol starts, from some interlocutor of *B-role*). To formalise this, we will say that a variable is *free in* or *bound to a role* (see formalisation below).

Variables in a Role. A variable x is *bound* to a role R, written $x \in \mathcal{B}_R$, if the (CAPSL) protocol description stipulates the variable x as private to the role R. Otherwise, a variable y is *free* in a role R, written $y \in \mathcal{F}_R$. The extension to non-atomic terms is as usual: if $t \in T_{\Sigma,X'}$ and $X' \cap \mathcal{F}_R \neq \emptyset$ then $t \in \mathcal{F}_R$, otherwise $t \in \mathcal{B}_R$.

In the Pr_1 protocol the variable n is bound to the role of A, while the variable m is free in the role of A. Therefore, the term $\leq(m,n)$ is free in the role of A.

To denote the initial ignorance of some concrete values within a role, we will use designated values, called *null values*, in the assignment of the variables and terms which are free in a role. To denote these null values, we use $(\perp_s \,|\, s \in S)$, an S-sorted set of constant function symbols. When the sort s is implicit, we simply write $\perp$ instead of $\perp_s$. All constant function symbols over $\omega \in S^*$ in which at least one component is $\perp$ are denoted by $\perp_\omega$, i.e., if n has value $\perp_{nat}$ within $\leq(m,n)$, then the whole value of $\leq(m,n)$ is also $\perp$, specifically $\perp_{[(nat,nat),bool]}$. Bound variables are assigned to concrete, non-null values, chosen arbitrarily over the a given range, e.g., integers, etc. Let the universal algebra $\mathbb{A}$ be the denotational interpretation of the theory (Σ, E). To be able to define operations on null values, we naturally extend the denotation $\mathbb{A}$ to $\mathbb{A}_\perp$, which has $A_{s_\perp} = A_s \cup \{\perp_s\}$ as support-sets and it operates over A_s like $\mathbb{A}$ and it returns $\perp$ whenever it operates over $\perp$, for any $s \in S$.

Initial Instantiation of Roles. Let $\delta \in \Delta$ be an assignment. The *initial R-role instantiation* $\delta|_R$ is the projection of the assignment δ on a role R, extended to $\mathbb{A}_\perp$ to enforce the assignment of all terms in the *R-role* to values, including null values: $\delta|_R = (t/_{\delta(t)}, t'/_{\perp_s} \,|\, t \in (\mathcal{B}_R)_s, t' \in (\mathcal{F}_R)_s, s \in S)$.

For the protocol Pr_1, let $\delta = [n/_3, m/_2, A/_{alice}, B_{bob}]$ be an assignment. Then, by the definition above, the initial role instantiations are as follows:
$\delta|_A = [n/_3, m/_\perp, A/_{alice}, B/_{bob}, \leq(n,m)/_\perp]$,
$\delta|_B = [n/_\perp, m/_2, A/_{alice}, B/_{bob}, \leq(n,m)/_\perp]$.

For each role R, we map each initial instantiation $\delta|_R$ of the *R-role* into an agent ag_R^δ. This gives the *set* $Ag = \bigcup_{\delta\in\Delta} \bigcup_{R \in X_{role}} \{ag_R^\delta\}$ *of agents.*

An IS Protocol Semantics.

In the following let the agent ag_R^δ represent an arbitrary *R-role* under the assignment δ of a *Pr* protocol. In particular, let ag_A^δ correspond to the *A-role* under $\delta|_A$, for an assignment $\delta = [n/_3, m/_2, A/_{alice}, B/_{bob}]$ of the protocol Pr_1. We now present the formal description of the agent ag_R^δ.

We consider several concurrent instantiations of each role by different agents. So, a free term ($\perp$) representing the role of the sender can later be instantiated to potentially different values, depending on the value received from other agents. A receipt may trigger the instantiations of other local terms as prescribed by the equational theory of the protocol. For instance, in Pr_1 with $\delta = [n/_3, m/_2, A/_{alice}, B/_{bob}]$ an *A-role* participant may receive the value 2 for m from a *B-role* agent. Following this, the *A-role* agent will "apply" the equational theory E_1 to rewrite the term $\leq(m,n)$ to $\leq(3,2)$, $\leq(2,1)$, $\leq(1,0)$ and, finally, to *.F.*[1]. To permit this, the term $\leq(m,n)$ of sort *Bool*, which is free in the role of A, should range over $(A_{nat} \times A_{nat}) \cup A_{bool_\perp} = (\mathbb{N} \times \mathbb{N}) \cup \{.T., .F.\}_\perp$. However, the term n should efficiently range only over $A_{nat} = \mathbb{N}$ for this agent, since n is bound to the *A-role* and its initial value cannot be changed. These value-range restrictions optimise the size of the fully instantiated model. The following definition formalises this by giving the possible values of a portion of a message held by an agent during the run.

Range of a Term for an Agent. The *range* $Range_R(t)$ *of a term* $t \in T_{\Sigma,X}|_{Pr}$ *for an* ag_R^δ *agent* is as follows:
$Range_R(t)=$

$$\begin{cases} A_s & t \in (\mathcal{B}_R)_s \cap X_s \quad (\mathbf{1})\\ A_{s_\perp} & \text{if } t \in (\mathcal{F}_R)_s \cap X_s \quad (\mathbf{2})\\ (A_{s_1} \times \ldots \times A_{s_n}) \cup A_s & \text{if } t \in (\mathcal{B}_R)_s,\ t = \sigma(t_1,\ldots,t_n),\\ & \sigma \in \Sigma_{(s_1,\ldots,s_n),s},\ t_i \in T_{\Sigma,X,s_i} \quad (\mathbf{3})\\ (A_{s_1} \times \ldots \times A_{s_n}) \cup A_{s_\perp} & \text{if } t \in (\mathcal{F}_R)_s,\ t = \sigma(t_1,\ldots,t_n),\\ & \sigma \in \Sigma_{(s_1,\ldots,s_n),s},\ t_i \in T_{\Sigma,X,s_i} \quad (\mathbf{4}) \end{cases}$$

Stores and Views for an Agent. A *store for an agent* ag_R^δ is a relation $(t :: Range_R(t) \,|\, t \in T_{\Sigma,X}|_{Pr})$ between terms and their respective ranges for the agent ag_R^δ.

An *initial view for an agent* ag_R^δ encodes an a initial R-role instantiation δ. A *non-initial view for* ag_R^δ encodes an actual assignment $\delta[y/_v]$, for some $y \in (\mathcal{F}_R)_s$, $v \in A_s$ (i.e., $v \neq \perp_s$).

The store of ag_A^δ in a model for Pr_1 is as follows:
$store_{ag_A^\delta} = (A :: String, B :: String, n :: \mathbb{N}, m :: \mathbb{N}_\perp,$
$\leq(n,m) :: (\mathbb{N} \times \mathbb{N}) \cup \{.T., .F.\}_\perp)$.

A possible non-initial view for ag_A^δ in a model for Pr_1 is:
$view_{ag_A^\delta} = (A \mapsto alice, B \mapsto bob, n \mapsto 3, m \mapsto 2,$
$\leq(n,m) \mapsto \perp)$.

The non-initial view $view_{ag_A^\delta}$ shows that ag_A^δ has updated the value for m in its view from the initially held $\perp$ to the received value 2. In the above, ag_A^δ has not yet "calculated" the value of $\leq(n,m)$, i.e., $\leq(n,m)$ is still $\perp$ in $view_{ag_A^\delta}$.

[1] *.T.* and *.F.* are the concrete values for *true* and *false*.

To complete the description of instantiated protocol roles, we introduce the set of *adjacent terms* Adj. These terms are unrelated to the equational theory E, but are induced by the (CAPSL) protocol description (e.g., flag variables to denote protocol steps, stages of rewriting, etc.).

Local States of Agents. An *(initial) local state* is an (initial) view together with certain protocol-driven *adjacent* terms and their assigned values. The set $L_{ag_R^\delta}$ is the set of all *possible local states* of ag_R^δ.

Let $step \in Adj$ be an adjacent atomic term of sort nat (i.e., denoting protocol steps). Then, let $Range_R(step) = \mathbb{N}$ and $step \mapsto 1$ in an initial setup, for any role R. An initial state i_l and local state l of agent ag_A^δ in a model for Pr_1 are as follows:
$i\_l = (step \mapsto 1, A \mapsto alice, B \mapsto bob, n \mapsto 3, m \mapsto \bot_{\mathbb{N}},$
$\quad \leq(n,m) \mapsto \bot_{\{.T.,.F.\}});$
$l = (step \mapsto 5, A \mapsto alice, B \mapsto bob, n \mapsto 3, m \mapsto 2,$
$\quad \leq(n,m) \mapsto .F.).$

Let $l \in L_{ag_R^\delta}$, $t \in T_{\Sigma,X}$ and $x \in Range_R(t)$. In the following, we use the following notations:

$l.view$ denotes the view encoded inside the local state l
$l.t$ denotes that the local state l stores a value for the term t
$l\|_{t=x}$ denotes the fact that $l.t{=}x$
$l\|_\delta$ denotes that $l\|_{t=x}$ and $\delta{=}[t/_x]$ for all t in the domain of δ
$l[t/_x]$ denotes the fact that $l.t$ is set to x

In the following, let i denote the map ag_R^δ of an initiator role R and i' denote the map $ag_{R'}^{\delta'}$ of a receiver role R'.

Local Actions and Protocol of Agents. Let $step \in Adj$, $j \in \{1,2,3\}$, $nr_j \in Range_R(step)$ and t, x, i', l be as above. The set $LAct_i{=}\{send(t,x,i'), receive(t,x), rewrite, empty\}$ is the set of *possible local actions* of agent i. The *local protocol* P_i of agent i is as follows: $P_i(l|_{step=nr_1,\, l.t=x,\, l.R'=i'.R'}){=}\{send(t,x,i')\}$, $P_i(l|_{step=nr_2}){=}\{receive(t,x)\}$, $P_i(l|_{step=nr_3}){=}\{rewrite\}$.

When a particular protocol is given, the parameters of the actions are restricted to proper subsets of $T_{\Sigma,X}$, e.g., t ranges over certain terms in $receive(t,x)$ for i.

The Environment Agent. We assume that the environment agent Env records all communication. Therefore, the local states of the Env agent are given by maps of the form $(t :: \bigcup_{R\in role} Range_R(t) :: Ag :: Ag \mid t \in T_{\Sigma,X})$. This gives the set L_{Env} of *possible local states of the Environment agent.* The environment has only one possible action denoted by *listen*, which is enabled by its protocol at every local state.

Global States and Joint Actions. Let $i \in Ag = \{1,\dots,n\}$, $l_i \in LAct_i$, $l_{Env} \in L_{Env}$, $a_i \in Act_i$ and $a_{Env} \in Act_{Env}$. A *global state* g is a tuple $(l_1,\dots,l_n,l_{Env})$. The set G of global states is the set of all possible states g as above. A *joint action* a is a tuple $(a_1,\dots,a_n,a_{Env})$. The set Act of joint actions is the set of all possible joint actions a as above.

Agents' Local Evolution Function. Let i denote the ag_R^δ agent, i' denote the $ag_{R'}^{\delta'}$ agent as above, let $l \in L_i$ be a local state of agent i and $a \in Act$ be a joint action. The *local evolution function* E_i of agent i is defined below. In this definition, the preconditions for enabling a state-update upon receipt express the following: *1*) the action *receive* of agent i is synchronised with the action *send* of agent i' and with the action *listen* of the Env agent; *2*) agent i is in the step nr where it awaits message t; *3*) the purported sender is the agent of the R'-*role*[2] (i.e., $i.R' = i'.R'$); *4*) the values x_j of certain subterms t_j in the received term t are consistent with agent i's view, i.e., $l|_{t_j=x_j}$. These conditions are inspired by the matching-receive semantics [3, 21].

$$\begin{cases} l[step/_{nr+1}] & \text{if } l|_{t=x,step=nr,R'=i'.R'}, \text{ for} \\ & a_i{=}send(t,x,i'), a_{Env}{=}listen, \\ & a_i' = receive(t,x) \\ l[step/_{nr+1}, t/x] & \text{if } l|_{t_j=x_j,step=nr,R'=i'.R'}, \text{ for} \\ & a_i{=}receive(t,x), a_{Env}{=}listen, \\ & a_i'{=}send(t,x,i),\ t_j \in Sub(t) \\ l[step/_{nr+1}, t/t'] & \text{if } l|_{step=nr+1}, \text{ for } a_i{=}rewrite, \\ & a_{Env}{=}listen, \\ & a_i'{=}empty,\ t \in T_{\Sigma,X}, t' = t{\downarrow_E} \end{cases}$$

To illustrate further, let $i = ag_A^\delta$ in the Pr_1 protocol and, by E_i, let the action $receive(m,2)$ be performed at the local state $l|_{step=1}$ of agent i. The implicit rewriting-driven state-update is: $\leq\!(3,2) \to \leq\!(succ(2), succ(1)) \to \leq\!(2,1) \to \leq\!(succ(1), succ(0)) \to \leq\!(1,0) \to .F.$. For protocols where the intermediate rewriting is not of interest, we collapse such a state-update sequence in one update, i.e., the sequence $l[step/_3, \leq(n,m)/_{(2,1)}]$, $l[step/_4, \leq(n,m)/_{(1,0)}]$ and $l[step/_5, \leq(n,m)/_{.F.}]$ is reduced to $l[step/_3, \leq(n,m)/_{.F.}]$. The above presentation of the local evolution function E_i formalises such optimisations.

The Global Evolution Function. The *global evolution function* $t : G \times Act \to G$ is such that $t(g,a) = g'$ if $act_i \in P_i(g_i)$, $E_i(g_i,a) = g_i'$, for all $i \in Ag \cup \{Env\}$, for $g, g' \in G$ and $a \in Act$.

A *path* is a sequence of global states described by the global evolution function. Paths naturally define the set of *reachable* states. Henceforth, G refers to the set of reachable states.

Equational Interpreted System for Pr. An *equational interpreted system for Pr*, denoted by Υ_{IS}^E, is a tuple $\mathcal{I}{=}(G, Act, P, t, I_0, V)$, where the components Act and t are as previously defined, $I_0 \subset G$ is a set of initial global states, $P = (P_i \mid i \in Ag \cup \{Env\})$, and $V : G \times PV \to \{true, false\}$ is a valuation function for the propositions PV of a logic language.

Local Satisfaction of Equational Equalities of Terms. The local state $l \in L_i$ *satisfies* $t =_E t'$, written $l{\models}(t =_E t')$, if $l|_{t=t'}$, for $t \to_{E*} t'$ with $t' = t{\downarrow_E}$, $t \in T_{\Sigma,X}|_{Pr}$ (i.e., $t \notin Adj$).

By the definition above, a local state l satisfies the equality $t =_E t'$ of terms modulo E if the term t has been rewritten to the normal term t' in local state l of Υ_{IS}^E.

Equational Indistinguishability. Two local states $l \in L_i$ and $l' \in L_i$ are *i-indistinguishable modulo E*, written $l \approx_i^E l'$, if it is the case that $l{\models}(t =_E t')$ if and only if $l'{\models}(t =_E t')$, for all $t \in T_{\Sigma,X}|_{Pr}$, i.e., $t \notin Adj$. Two reachable global states $g, g' \in G$ are *i-indistinguishable modulo E*, written $g \sim_i^E g'$, if $g_i \approx_i^E g_i'$. The relation $\sim_i^E \subseteq G \times G$ is the *quotient-indistinguishability relation.*

By the definition above, two local states are indistinguishable modulo E if they satisfy the same equalities of terms modulo E.

[2]If protocols use anonymous channels, then this condition is dropped.

As an example, let i be ag_A^δ in the $\Upsilon_{IS}^{E_1}$ for the protocol Pr_1. As none of the following states of ag_A^δ satisfy $\leq(m,n) =_{E_1} .F.$, it holds that $l_1|_{step=2,\leq(n,m)=(3,2)} \approx_i^{E_1} l_2|_{step=3,\leq(n,m)=(2,1)} \approx_i^{E_1} l_3|_{step=4,\leq(n,m)=(1,0)}$. However, the state $l|_{step/5,\leq(n,m)/.F.}$ does satisfy $\leq(m,n) =_{E_1} .F.$

Equational Multi-agent System Model for *Pr*. Let $\mathcal{I}$ be an equational interpreted system for a protocol *Pr* specified by a convergent equational theory (Σ, E). The *equational multi-agent system model for Pr* $M_{IS}^E = (G, (\sim_i^E)_{i\in Ag}, V)$ is the model generated by the equational interpreted system model $\mathcal{I}$. In M_{IS}^E the relation $\sim_i^E$ is as described above, G is the set of reachable states generated by $\mathcal{I}$, and V is the set of atomic proposition in $\mathcal{I}$.

We use the notation $\mathcal{I}$ both for the equational interpreted system for *Pr* and the equational multi-agent system model for *Pr*; the context will disambiguate. In our implementation, we optimise the formalism above when generating the Υ_{IS}^E; this is not discussed here.

2.3 The Epistemic Logic CTLKR

Let $\mathcal{I}$ be the equational multi-agent system model M_{IS}^E of *Pr*, $p \in PV$ and $i \in Ag \cup \{Env\}$. The specification language CTLKR, used to express the system requirements is defined by the following BNF:

$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i\varphi \mid R_i\varphi \mid AX\varphi \mid AG\varphi \mid A(\varphi U \varphi)$.

The operator K_i denotes the *knowledge modality* ($K_i\varphi$ reads "agent i knows the fact φ") while the operator R_i is the *rewriting-knowledge modality* ($R_i\varphi$ reads "agent i knows the fact φ modulo the equational theory (Σ, E)"). The semantics for CTL on M_{IS}^E is as on standard interpreted systems [12]. The interpretation of the knowledge modalities is as follows:

$$(\mathcal{I}, g) \models K_i\varphi \quad \text{if} \quad (\text{for all } g' \in G)(g_i = g'_i \text{ implies } (\mathcal{I}, g') \models \varphi)$$
$$(\mathcal{I}, g) \models R_i\varphi \quad \text{if} \quad (\text{for all } g' \in G)(g \sim_i^E g' \text{ implies } (\mathcal{I}, g') \models \varphi).$$

The logic CTLKR extends the commonly used logic CTLK by means of the rewriting epistemic modality R.

3. AN APPROXIMATION FOR AUTOMATIC VERIFICATION

We wish to use the logic CTLKR as a specification language for model checking security protocols encoded in the MAS-based formalism presented in the previous section. This would enable us to surpass the significant limitations of the state-of-the-art as discussed in the introduction. However, locally parametrised properties of type $t =_E t'$ make the computation of the indistinguishability relation particularly costly, thereby increasing the verification time. To circumvent this, we approximate the R modality and interpret it over an abstraction of Υ_{IS}^E through the use of local predicates. In the following we will show that important classes of protocols are amenable to analysis through this approximation.

3.1 Empirical Interpreted Systems

An S-*sorted logical signature* contains *logic symbols* of type $[\omega]$, for $\omega \in S^*$. Informally, a (standard) signature specifies symbols related to algebraic operators, e.g., *decrypt*, whereas a logical signature specifies symbols related to facts, e.g., *isDecrypted*.

Logically Extended Signatures. A *logically extended signature* is given by a tuple (Σ, Σ_L), where Σ is an S-sorted signature and Σ_L is an S-sorted logical signature. The tuple (Σ, Σ_L, E) is the *logically extended equational theory* corresponding to the equational theory (Σ, E). A logically extended equational theory can describe more properties of a protocol than the underlying equational theory alone.

The set $T_{\Sigma,\Sigma_L,X}$ of *logical terms* is defined on logically extended signatures (Σ, Σ_L) in the same way the set $T_{\Sigma,X}$ of terms is defined on the signature Σ.

The denotation of logically extended signatures is given through a *logical extension of the algebra* $\mathbb{A}_\perp$. In this extension the *interpretation* $i\_p^{\mathbb{A}_\perp}(\delta)$ *of a logical term* $p \in T_{\Sigma,\Sigma_L,X}$ *under assignments* $\delta \in \Delta$ is a predicate $p^{\mathbb{A}_\perp}$ evaluated over $\{true, false\}$. When $\mathbb{A}_\perp$ is implicit, we simply write $i\_p(\delta)$ instead of $i\_p^{\mathbb{A}_\perp}(\delta)$.

Let j be an arbitrary agent in an IS formalisation.

Logical Terms and Experiments of Agents. A fixed set $In_j \subseteq T_{\Sigma,\Sigma_L,X}$ denotes *the set of logical terms of agent* j. The set $InEx_j = \{i\_p(\delta) \mid p \in In_j\}$ of predicates contains the *local experiments for agent* j. An $InEx_j$ set is denoted as a *local experiment-set*. The Ag-indexed set $InEx = (InEx_j \mid j \in Ag)$ is the *experiment-set*.

Logical terms are symbols that enrich the agents' stores with "meta-data" representing facts not explicitly included in the protocol. Experiments are predicates that are evaluated on the views or the local states of agents, i.e., interpreting this meta-data. Thus, evaluating these predicates will account for a special kind of knowledge accrued by the agents.

Below we illustrate these notions, with intuitive predicates and symbols: e.g., $i\_\text{diffOne}(n,m)(\delta)$ is *true* if "the absolute difference between $\delta(n)$ and $\delta(m)$ is 1".

Possible Experiments For Agent ag_A^δ *in* Pr_1. The sets In of logical terms and their respective experiment-sets $InEx$ are as follows:

- $In1_i = \{\text{smaller}(n,m)\}$; $InEx1_i = i\_\text{smaller}(n,m)(\delta)$;
- $In2_i = \{\text{diffOne}(n,m)\}$; $InEx2_i = i\_\text{diffOne}(n,m)(\delta)$.

Similarly to [17], we introduce an indistinguishability relation defined over local predicates, i.e., here on local experiments of agents.

Local Empirical Indistinguishability. Two local states $l, l' \in L_j$ are *indistinguishable modulo* $InEx_j$, or $l \approx^{InEx_j} l'$, if $i\_p(\delta) = i\_p(\delta')$ for all $p \in In_j$, where $\delta, \delta' \in \Delta$ respectively describe l and l', i.e., $l|_\delta$ and $l'|_{\delta'}$. Two global states $g, g' \in G$ are *indistinguishable modulo* $InEx_j$, written $g \sim^{InEx_j} g'$, if $g_j \approx^{InEx_j} g'_j$. The relation $\sim^{InEx_j} \subseteq G \times G$ is the *empirical indistinguishability* relation.

Then, two local states l and l' are indistinguishable through experiments if these are evaluated identically at l and at l', as exemplified below.

Empirical Indistinguishability in a Model for Pr_1. Let i be the agent ag_A^δ in a model for Pr_1 and two local states of ag_A^δ respectively described by $\delta[n/_9, m/_8]$ and $\delta[n/_9, m/_5]$, i.e., $l_{ag_A}|_{\delta[n/_9,m/_8]}$ and $l'_{ag_A}|_{\delta[n/_9,m/_5]}$. Let $InEx1_i$ and $InEx2_i$ be the experiment-sets above. Then,

- $i\_\text{smaller}(n,m)(\delta[n/_9, m/_8]) = false$ and $i\_\text{smaller}(n,m)(\delta[n/_9, m/_5]) = false$;
- $i\_\text{diffOne}(n,m)(\delta[n/_9, m/_8]) = true$ and $i\_\text{diffOne}(n,m)(\delta[n/_9, m/_5]) = false$.

Therefore, $l \approx^{InEx1_i} l'$ holds, but $l \approx^{InEx2_i} l'$ does not hold, i.e., $l \not\approx^{InEx2_i} l'$.

Therefore, we have just augmented the Υ_{IS}^E formalisation of protocol executions with local experiments. The resulting models are formally defined below.

Empirical Equational Interpreted System. Let Pr be a protocol specified by (Σ, Σ_L, E) and $\mathcal{I}$ be the equational interpreted system Υ_{IS}^{E} for Pr. An *empirical equational interpreted system* is the tuple $\mathcal{I}'{=}(\mathcal{I}, InEx)$, where $InEx = (InEx_j \mid j \in Ag \cup \{Env\})$.

3.2 Extended Protocol Logic

We use P_i as the *empirical knowledge modality* ($P_i\varphi$ reads "agent i empirically knows the fact φ"). On an empirical equational multi-agent system model $\mathcal{I}' = (\mathcal{I}, InEx)$, consider the language $\mathcal{L}' = CTLKR \cup P_i$, for $i \in Ag \cup Env$.

The language CTLKR is interpreted on $\mathcal{I}'$ as on $\mathcal{I}$. Let $g \in G$ be an arbitrary reachable state of $\mathcal{I}'$. The interpretation of the empirical knowledge modality is as follows:

$$(\mathcal{I}', g) \models P_i\varphi \text{ if}$$
$$\text{(for all } g' \in G)(g \sim^{InEx_i} g' \text{ implies } (\mathcal{I}', g') \models \varphi).$$

The empirical knowledge of an agent refers to the information obtained only by theoretically enquiring the agent's local predicates, i.e., its experiments.

3.3 Experiments Sets of Convergent Equational Theories

In this section we explicitly link the convergence of the underlying equational theory to the experiments of the agents. Once again, we assume that the normal terms of the theory are encoded in the model, i.e., by using a rewriting system, and that the number of protocol instantiations considered is bounded.

Let Pr be a protocol specified by a *convergent* equational theory (Σ, E), $\mathcal{I}$ be the equational interpreted system for Pr and j denote the ag_R^{δ} agent as before. Let Σ_L be a logical signature containing the (special) logical symbols $pred \in \Sigma_L$ of type ω, for all $\omega \in S^*$. Let t be an arbitrary term of type ω, i.e., $t{\in}T_{\Sigma,X}$.

Predicates for Terms. A logical term $pred(t) \in T_{\Sigma,\Sigma_L,X}$ is a *logical term for* $t \in T_{\Sigma,X}$. The *interpretation* $i\_pred^E(t)(\delta)$ *of a predicate for* t *in* E is always *true*, i.e., $i\_pred^E(t)(\delta) = true$, for all $\delta{\in}\Delta$.

By the definition above, a predicate $i\_pred^E(t)$ for a term $t \in T_{\Sigma,X}$ is *true* under all assignments $\delta \in \Delta$ for t. Since δ is in Δ, i.e., δ is not an role instantiation, it means that $\delta(t) \neq \bot$. In the next definition we use predicates for terms to express special experiments, which simulate the recording of the normal terms.

Local Experiments of Convergent Theories.
$In_j^E{=}\bigcup_{t \in T_{\Sigma,X}} \{pred(t') \mid t' = t\downarrow_E\}$ is the set of logical terms *for the convergent theory* E of agent j. $InEx_j^E = \{i\_pred^E(t)(\delta) \mid pr(t){\in}In_j^E\}$ is set of local experiments *of the convergent theory* E for agent j.

Importantly, one can automatically produce the exact set of experiments of a convergent theory E for a protocol Pr by using the CAPSL description of the protocol, the finite set of instantiations given and the normal terms implied by E.

Empirical Equational IS for Convergent Theories. Let $InEx_j^E$ be the local experiments *for the convergent theory* E of agent j and $InEx^E{=}(InEx_j^E \mid j \in Ag)$. An empirical equational interpreted system $\Upsilon_{IS}^{\mathbb{I}E}$ *for the convergent theory* E is given by the tuple $\mathcal{I}'{=}(\mathcal{I}, InEx^E)$.

The unwinding of $\Upsilon_{IS}^{\mathbb{I}E}$ follows as in previous definitions. By the above, the system $\Upsilon_{IS}^{\mathbb{I}E}$ is a special empirical system, i.e., agents "track" normal terms under E. We now prove that in these systems the P_i modality coincides with R_i.

Theorem 3.1. *Let Pr be a protocol specified by a convergent theory (Σ, E) and $\mathcal{I}$ be an $M_{IS}^{\mathbb{I}E}$ model for E. Then, $\mathcal{I}{\models}\varphi$ if and only if $\mathcal{I}{\models}\overline{\varphi}$, for any $\varphi \in \mathcal{L}$, where $\overline{\varphi} \in \mathcal{L}'$ is obtained from φ by uniformly substituting R_j for P_j, for any $j \in Ag$.*

Proof (sketch). We only need to prove that $\mathcal{I}{\models}R_j\psi$ iff $\mathcal{I}{\models}P_j\overline{\psi}$, for some arbitrary $\psi \in \mathcal{L}$ and an agent $j{=}ag_R^{\alpha}$ under an initial R-role instantiation α.

Thus, $\mathcal{I}{\models}R_j\psi \overset{\text{def. of}\models\mathcal{L}}{\Leftrightarrow}$ (for all $g' \in G$)($g \sim_j^E g'$ implies $(I, g'){\models}\psi) \overset{\text{def. of}\approx^E, \models\text{eq}}{\Leftrightarrow}$ (for all $g' \in G$) (for all $t \in T_{\Sigma,X}, t' = t\downarrow_E$) $((g_j|_{t=t'}$ iff $g'_j|_{t=t'})$ implies $(\mathcal{I}, g'){\models}\psi)$ (**1**). Let δ, δ' be assignments extending the initial R-role instantiation α and w.l.o.g. denote the local states in (1) as $g_j|_\delta,\ \ g'_j|_{\delta'}$ (**2**). Then, $In_j^E{=}\bigcup_{t \in T_{\Sigma,X}} \{pred(t\downarrow_E)\}$, $InEx_j^E = \cup_{t \in T_{\Sigma,X}} \{i\_pred(t\downarrow_E)(\delta) = true\}$ (**3**). By the definition of $\approx^{InEx_j^E}$, (2) and (3), the following holds in (1): $g_j \approx^{InEx_j^E} g'_j$ (**4**). From (1) with the above and (4), it follows that: $\overset{\text{def. of}\models\mathcal{L}'}{\Leftrightarrow}$ $(\mathcal{I}, g){\models}P_j\psi$. $\square$

4. MODEL CHECKING KNOWLEDGE OF PROTOCOL PARTICIPANTS

In this section we present a procedure for model checking empirical knowledge that allows for the specification and verification of standard interpreted systems equipped with local experiment-sets, i.e., not only for the equationally-driven $\Upsilon_{IS}^{\mathbb{I}E}$.

Algorithm 1 SAT$_{\text{P}}$(φ : Formula, j : Agent) : *Set of* States

```
1: $X \leftarrow [\![\neg\varphi]\!]$
2: $Y \leftarrow X$
3: while $X \neq \emptyset$ do
4:     $g \leftarrow X$.pop()
5:     $\phi_g \leftarrow$ true
6:     for $exp \in InEx_j$ do
7:         if $g \in [\![exp]\!]$ then
8:             $\phi_g \leftarrow \phi_g \wedge exp$
9:         else
10:            $\phi_g \leftarrow \phi_g \wedge \neg exp$
11:        end if
12:    end for
13:    $Y \leftarrow Y \cup [\![\phi_g]\!]$
14:    $X$.remove($[\![\phi_g]\!]$)
15: end while
16: return $\neg Y$
```

The approach for calculating the set $[\![P_j\varphi]\!]$, i.e., the set of states that satisfy the formula $P_j\varphi$, is shown in Algorithm 1. Lines 8 and 10 construct the formula ϕ_g representing the conjunction of the evaluation of experiments for the agent j at the current state g. The set Y is constructed iteratively from each $g \in [\![\neg\varphi]\!]$ (the set X). At Line 13, $[\![\phi_g]\!]$ contains the set of states that are empirically indistinguishable from the state g (i.e., $[\![\phi_g]\!] = \{g' \in G \mid g' \sim^{InEx_j} g\}$). To calculate $Y = \{g \in G \mid (\exists g' \in G)(g' \sim^{InEx_j} g) \wedge (g \models \neg\varphi)\}$ efficiently, we remove $[\![\phi_g]\!]$ from X (Line 14) as these states have an identical experiment-set evaluation. At Line 15, Y contains

Table 1: E-Voting Specifications in CTLKR

VP	$AG(votes(i,v) \rightarrow AG \bigwedge_{v' \neq v} Q_{at}(votes(i,v')))$
VVU	$AG(votes(i,v) \rightarrow \bigwedge_{i' \neq i} \bigwedge_{v' \neq v} [votes(i',v') \rightarrow AGQ_{at}(votes(i,v') \wedge votes(i',v))])$
RF	for $v \in Range_V(vote) \setminus \{v_r\}$, $AG(votes(i_r,v_r) \wedge votes(i,v) \rightarrow \bigwedge_{i' \notin \{i,i_r\}} \bigwedge_{v'} [votes(i',v') \rightarrow AGQ_{at}(votes(i,v_r) \wedge votes(i',v) \wedge votes(i_r,v'))])$
CR	for $v \in Range_V(vote) \setminus \{v_c\}$, $AG(votes(i_c,v_c) \wedge votes(i,v) \rightarrow \bigwedge_{i' \notin \{i,i_c\}} \bigwedge_{v'} [votes(i',v') \rightarrow AGQ_{at}(votes(i,v_c) \wedge votes(i',v) \wedge votes(i_c,v'))])$

the reachable states that either directly refute φ, or are empirically indistinguishable from a state that does. Therefore we obtain that $Y = [\![\overline{P_j}(\neg\varphi)]\!]$, where $\overline{P_j}(\neg\varphi) \equiv \neg P_j(\neg\varphi)$ is the dual of $P_j(\varphi)$. Finally, at Line 16, the algorithm calculates $\neg Y$, i.e., the set difference between the set of global states G and Y. So, it returns $[\![P_j(\varphi)]\!]$.

PROPOSITION 4.1. *Algorithm 1 calculates the set of states* $[\![P_j\varphi]\!]$.

Implementation. We have implemented Algorithm 1 as an experimental extension of the model checker `MCMAS` [15]. This extension, titled `MCMAS-E`, is available from [1]. ISPL, The input-language of `MCMAS`, was extended to allow for the definition of experiments at the agent level, as well as to support the specification of empirical knowledge formulae.

5. VERIFYING E-VOTING PROTOCOLS

The applicability of previous research [3] in this line has been limited to protocols for which the specifications can be expressed by using standard notions of knowledge; this included authentication and key-establishment. We herein analyse e-voting protocols, which were out of the scope of [3].

To illustrate that the models and knowledge modalities introduced so far surpass this limit, we analyse more sophisticated e-voting protocols than previously possible with our extension of `MCMAS`.

E-Voting in the $\Upsilon_{IS}^{\mathbb{I}E}$ Formalism. Assume a $\Upsilon_{IS}^{\mathbb{I}E}$ model and the propositions *votes*(j) and *votes*(j,x), representing that an honest agent j has voted and that agent j has voted x, respectively. Let i, i' be two different agents. We consider only fair paths representing voting sessions in which eventually both agent i and agent i' vote and that the voting is not unanimous.

The specifications for e-voting requirements we consider, i.e., *vote privacy* (VP), *voter-vote unlinkability* (VVU), *receipt-freeness* (RF) and *coercion-resistance* (CR), are formalised in Table 1. We use the notation $Q_j\varphi$ to represent $\neg R_j \neg\varphi$, for any agent j.

VP stipulates that whenever agent i has voted v, there does not exist a point where the attacker at can be sure that it was i who voted v. Similarly, *VVU* expresses that the attacker at will always consider it possible that agents i and i' have swapped votes. *RF* states that, whenever agent i counterbalances the vote of the receipt-providing agent i_r, the attacker at is not at any point able to link any of the voters to their respective votes. *CR* is similar to RF, but it is analysed on a stronger threat-model. The formulae VVU, RF and CR are inspired by the specifications of *total role-interchangeability* [23], whereas VP is inspired by the specifications of anonymity in [14] and their extensions to privacy in [23].

We verify these specifications against the FOO'92 e-voting protocol [13]. We formalise the execution of a finite number of concurrent sessions as three, specialised $\Upsilon_{IS}^{\mathbb{I}E}$ systems. The first model, $\mathcal{M}_1$, is a $\Upsilon_{IS}^{\mathbb{I}E}$ model with an added *Attacker* agent (at) representing a passive intruder. This model satisfies the vote-privacy property. A receipt-providing agent i_r and a stronger *Attacker* are modelled in $\mathcal{M}_2$, which specialises $\mathcal{M}_1$ and supports receipt-freeness. To model coercion, the formalisation $\mathcal{M}_3$ extends $\mathcal{M}_2$, with a further enhanced *Attacker* and a coercible agent i_c.

Experiments. The high-level description of the FOO'92 protocol was initially provided in CAPSL [9]. This encoding was then passed to an ISPL translator [1]. The translator is an extension of the PD2IS toolkit [3] where the instantiated, $\perp$-enhanced normal terms are inserted into the ISPL models. In this way we can automatically generate the $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$ formalisations of FOO'92, as well as the e-voting requirements in ISPL. The generated ISPL files are in the region of 8000 lines and take approximately 15 seconds to build. `MCMAS-E` was then used to verify these models. The machine employed was an Intel Core 2 Duo processor 3.00 GHz with a 6144 KiB cache running the 32-bit Linux kernel 2.6.32.10. The averaged results obtained across two runs of `MCMAS-E` are summarised in Table 2.

Table 2:Averaged Experiments on FOO'92.

	Form.	Mem. (KiB)	Time (s)	States
$\mathcal{M}_1$	VP/VVU	176032	66441	$6.69 \cdot 10^{11}$
$\mathcal{M}_2$	(weakened) RF	175496	66168	$6.69 \cdot 10^{11}$
$\mathcal{M}_3$	VP/VVU	181926	70401	$6.69 \cdot 10^{11}$

The leftmost column shows the class of model considered. The *Memory* and *Time* columns respectively show the average memory usage and the average CPU time in each run. *States* reports the number of reachable states in each model.

Discussion of Results. The *Formulae* column reports the strongest e-voting specification that was found to hold on the model (strength grows from VP, VVU, RF to CR); vote privacy (VP and VVU) were found to hold on all three classes of models considered. On models $\mathcal{M}_2$ and $\mathcal{M}_3$ a path was found where eventually the intruder is able to link the receipt-providing agent and its vote, i.e., receipt-freeness (RF) was refuted. Our findings are in-line with known results (i.e., vote-privacy holding for FOO'92). An alternative approach based on applied-pi [8] exhibits similar results. The models verified are of large sizes and are not optimised for e-voting, consequently our verification times are seen as favourable. The complete set of ISPL models and specifications verified are available from [1].

6. CONCLUSIONS AND RELATED WORK

In this paper we have introduced an approach to model checking MAS-based models of security protocols, using specifications expressed in a specialised temporal-epistemic logic. This work surpasses the current state-of-the-art of temporal-epistemic verification of protocols specified as MAS in two ways. Firstly, it advances a formalism integrating equational theories with epistemic logic. This allows for the modelling of several cryptographic primitives of interest. Secondly, we present an automatic methodology for the verification multi-agent systems-based models against relevant specifications through an open-source dedicated model checker. We emphasise nonetheless that the methodology presented is not directly optimised for e-voting primitives; in fact, it aims at a generic MAS-based verification method.

The empirical indistinguishability relation introduced is related to that of explicit knowledge [17], although in that line no support for cryptographic primitives was available. In [20] agents are empowered with deduction algorithms for generating new local knowledge, but the technical details are different and no automatic technique is discussed. In a theoretical setting of cryptographic modelling, [22] studied the decidability of model checking with respect to an epistemic extension of ATL*; given the specification language, it is clear that the protocol model, the operators and the semantics in [22] differ from those we present.

Semi-decidable tools have been used to show static equivalence of applied-pi frames modulo certain convergent equational theories [6, 8]. Such approaches could be applied to verify symbolically an infinite number of e-voting sessions. However, they focus mainly on the problem of deciding static equivalence in process calculi (thus a comparison on protocol verification cannot be drawn). Comparatively, we assume a bounded number of fully instantiated protocol sessions where the normal terms of the theory are encoded in the model. Thus, we attain a decidable and fully automatic method of MAS-based protocol verification. In the context of bounded size modelling, the epistemic modalities and indistinguishability relations we have introduced can be correlated to process equivalence [6, 8] and, respectively, static frame-indistinguishability in applied-pi calculus.

Our specifications of e-voting requirements follow the formulations of anonymity in [14, 23] and are model-independent (unlike those in [8], where e-voting specifications are expressed as reachability or process equivalence properties in a model-dependent manner).

Acknowledgments.

The authors acknowledge the support of EPSRC grants EP/E035655/1 and EP/I00520X/1, and the Swiss-funded MICS project.

7. REFERENCES

[1] MCMAS-E, Model Checking Equational IS. `http://vas.doc.ic.ac.uk/tools/mcmas_equational/`.

[2] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge Univ. Press, 1998.

[3] I. Boureanu, M. Cohen, and A. Lomuscio. Model Checking Temporal Epistemic Specifications for Authentication Protocols Compiled into Interpreted Systems. *Journal of Applied Non-Clasical Logics*, 19(4):463–487, 2009.

[4] I. Boureanu, A. Lomuscio, and M. Cohen. Model Checking Detectability of Attacks in Multiagent Systems. In *Proc. of AAMAS'10*, pages 691–698. IFAAMAS Press, 2010.

[5] M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990.

[6] Ş. Ciobâcă, S. Delaune, and S. Kremer. Computing Knowledge in Security Protocols under Convergent Equational Theories. In *Proc. of CADE'09*, pages 355–370. Springer, 2009.

[7] M. Cohen and M. Dam. A Complete Axiomatization of Knowledge and Cryptography. In *Proc. of LICS'07*, pages 77–88. IEEE Comp. Soc. Press, 2007.

[8] S. Delaune, S. Kremer, and M. Ryan. Verifying Privacy-Type Properties of Electronic Voting Protocols. *Journal of Computer Security*, 17(4):435–487, 2009.

[9] G. Denker and J. Millen. CAPSL Intermediate Language. In *Proc. of FMSP'99.* ACM New York, 1999.

[10] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics.* Springer, 1985.

[11] J. Ezekiel, A. Lomuscio, L. Molnar, S. Veres, and M. Pebody. Verifying Fault Tolerance and Self-Diagnosability of an Autonomous Underwater Vehicle. In *Proc. of IJCAI'11*, 2011.

[12] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge.* MIT Press, 1995.

[13] A. Fujioka, T. Okamoto, and K. Ohta. A Practical Secret Voting Scheme for Large Scale Elections. In *Proc. of AUSCRYPT'92*, pages 244–251. Springer, 1992.

[14] J. Halpern and K. O'Neill. Anonymity and Information Hiding in Multiagent Systems. *Journal Computer Security*, 13(3):483–514, 2005.

[15] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A Model Checker for Multi-Agent Systems. In *Proc. of CAV'09*, volume 5643, pages 682–688. Springer, 2009.

[16] A. Lomuscio, F. Raimondi, and B. Wozna. Verification of the TESLA Protocol in MCMAS-X. *Fundamenta Informaticae*, 79(3-4):473–486, 2007.

[17] A. Lomuscio and B. Woźna. A Combination of Explicit and Deductive Knowledge with Branching Time: Completeness and Decidability Results. In *Proc. of DALT'05*, volume 3904, pages 188–204. Springer, 2005.

[18] R. van der Meyden and P. Gammie. MCK: Model Checking the Logic of Knowledge. In *Proc. of CAV'04*, volume 3114 of *LNCS*, pages 479–483. Springer, 2004.

[19] R. Parikh and R. Ramanujam. Distributed Processes and the Logic of Knowledge. In *Logic of Programs*, pages 256–268, 1985.

[20] S. Petride and R. Pucella. Perfect Cryptography, S5 Knowledge, and Algorithmic Knowledge. In *Proc. of TARK'07*, pages 239–247. ACM Press, 2007.

[21] M. Rusinowitch and M. Turuani. Protocol Insecurity with Finite Number of Sessions is NP-complete. *Proc. of IEEE-S&P'01*, 2001.

[22] H. Schnorr. Deciding Epistemic and Strategic Properties of Cryptographic Protocols. Technical Report TR 1012, Christian-Albrechts-Universitat zu Kiel, Institut fur Informatik, October 2010.

[23] Y. Tsukada, K. Mano, H. Sakurada, and Y. Kawabe. Anonymity, Privacy, Onymity, and Identity: A Modal Logic Approach. In *Proc. of CSE'09.* IEEE.

Semantics and Verification of Information-Based Protocols

Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA

singh@ncsu.edu

ABSTRACT

Information-Based Interaction-Oriented Programming, specifically as epitomized by the Blindingly Simple Protocol Language (BSPL), is a promising new approach for declaratively expressing multiagent protocols. BSPL eschews traditional control flow operators and instead emphasizes causality and integrity based solely on the information models of the messages exchanged. BSPL has been shown to support a rich variety of practical protocols and can be realized in a distributed asynchronous architecture wherein the agents participating in a protocol act based on local knowledge alone. The flexibility and generality of BSPL mean that it needs a strong formal semantics to ensure correctness as well as automated tools to help develop protocol specifications.

We provide a formal semantics for BSPL and formulate important technical properties, namely, enactability, safety, and liveness. We further describe our declarative implementation of the BSPL semantics as well as of verifiers for the above properties using a temporal reasoner. We have validated our implementation by verifying the correctness of several protocols of practical interest.

Categories and Subject Descriptors

H.1.0 [**Information Systems**]: Models and Principles—*General*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Theory, Verification

Keywords

Business protocols, agent communication

1. INTRODUCTION

We take as our point of departure Singh's [13] recent work on Information-Based Interaction-Oriented Programming and especially on the Blindingly Simple Protocol Language (BSPL). The main innovation of BSPL is that it specifies multiagent protocols without the use of any control flow constructs. Instead, it relies purely on the specifications of the information schemas of the messages exchanged among the defined roles. From the message schemas, consisting of parameters (adorned in a specific manner we explain below), BSPL characterizes the relevant (1) causal dependencies and (2) integrity constraints. Causality provides a basis for ordering requirements that traditional languages address via control flow operators. Integrity constraints provide a basis for exclusion requirements that traditional languages address via choice operators.

Singh [13] explains the generality of BSPL in handling a variety of practical protocols and its ability to support the composition of protocols, but without violating encapsulation as previous approaches, e.g., AUML [11] and MAD-P [3], do. Further, Singh [14] shows how BSPL can be realized in a fully distributed, asynchronous architectural style wherein the participating agents can act based solely on local knowledge. The agents are always ready to receive any incoming message and are not prevented from emitting any message whose information prerequisites they can meet.

How can we be sure that a protocol is correct? Will it lead to erroneous enactments? Will it deadlock? Notice that BSPL merely forces the protocol designer to be explicit about causality, but it does not add to the challenges of correctness. Specifically, any approach that supports distributed decision making faces these problems. Traditional approaches insert arbitrary rigidness whereas BSPL offers an opportunity to achieve correctness and flexibility.

What is needed is, first, a rigorous formalization of the BSPL semantics that would capture the inherent distribution of a BSPL enactment along with expressing the local views of the (agents playing the) roles involved. Second, what is also needed is a clear formulation of important correctness properties of protocols and tools that would verify protocols with respect to those properties.

Contributions. The main contribution of this paper is precisely to fill the above gaps. To do so requires new technical results. One, we formulate a formal semantics for BSPL that incorporates a notion of *viability* and respects the locality of each role, the flow of causality across roles, and the asynchrony of the communications between them. Two, we capture the semantic requirements purely declaratively so that a logic-based reasoner can compute with them. Three, we formalize correctness properties. We realize the semantics and properties in a verification tool for BSPL protocols.

Organization. Section 2 follows Singh [13] in describing BSPL. Section 3 provides intuitions about the BSPL semantics as well as the correctness properties of interest. Section 4 provides a semantics of BSPL based on local enactments and observations by agents playing the roles in a protocol. Section 5 introduces a temporal language and shows how to formalize the causal structure of a protocol along with each property. It shows how we can verify each property by checking the (un)satisfiability of the conjunction of the causal structure and a property-specific formula. Section 6 discusses the related literature and some directions for future research.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

2. BACKGROUND ON BSPL

Listing 1 presents a protocol to help illustrate the main features of BSPL. For readability, in the listings, we write reserved keywords in sans serif, and capitalize role names. In the text, we write message and protocol names *slanted*, roles in SMALL CAPS, and parameters in sans serif. We insert ⌜ and ⌝ as delimiters, as in ⌜Self ↦ Other: hello[ID, name]⌝.

Listing 1: The *Purchase* protocol.

```
Purchase {
 role B, S, Shipper
 parameter out ID key, out item, out price, out
     outcome

 B ↦ S: rfq[out ID, out item]
 S ↦ B: quote[in ID, in item, out price]
 B ↦ S: accept[in ID, in item, in price, out
     address, out response]
 B ↦ S: reject[in ID, in item, in price, out
     outcome, out response]
 S ↦ Shipper: ship[in ID, in item, in address]
 Shipper ↦ B: deliver[in ID, in item, in address,
     out outcome]
}
```

BSPL distinguishes three main adornments on the parameters of a message: ⌜in⌝, meaning the binding must come from some other message; ⌜out⌝, meaning that the binding originates in this message (presumably based on private computations of the sender); and ⌜nil⌝, meaning that no binding is known to the sender at the time of emission. Each message instance must bind a proper value for each ⌜in⌝ and each ⌜out⌝ parameter, and a ⌜nul⌝ value for each ⌜nil⌝ parameter. For brevity, we avoid ⌜nil⌝ parameters in our examples.

Consider the quote message in *Purchase*, which includes an item description and a price, and may be emitted in response to a request for quotes for a particular item. Clearly, for the quote message to be emitted, its sender must instantiate all of its parameters. However, from the standpoint of the *quote* message, the item description is provided from the outside into the protocol and the price is provided by the protocol to the outside. Thus we adorn item with ⌜in⌝ and price with ⌜out⌝.

A message instance must provide a binding for each ⌜in⌝ and ⌜out⌝ parameter with the difference being that the ⌜out⌝ binding has declarative force [2]. For example, an agent emitting a price quote is not merely reporting a price previously computed in the conversation but declaring it to be the *definitive* price in this conversation. One can imagine such a message carrying the weight of a commitment, although we deemphasize commitments here.

All of *Purchase*'s parameters are adorned ⌜out⌝, indicating that *Purchase* provides them to any protocol that composes *Purchase*. The *rfq* and *quote* messages help generate a price offer. Here, the BUYER (B) generates item and the SELLER (S) generates price, since these parameters are adorned ⌜out⌝ in messages emitted by these roles. Any message that takes some ⌜in⌝ parameters can be enacted only if referenced from another protocol.

Notice that the *ship* message is irrelevant from the parameter standpoint since all its parameters are adorned ⌜in⌝, indicating that *ship* creates no new information. However, *ship* is clearly essential from the role perspective: it ensures that the SHIPPER learns of the parameter bindings that make the SHIPPER's emission of *deliver* viable. In general, BSPL separates and addresses the two concerns of the interplay of information with (1) interactions and (2) roles.

An enactment corresponds to a binding of public parameters. BSPL requires some of the parameters being declared as forming the *key* of a protocol enactment. Thus multiple concurrent enactments of the same protocol do not interfere with each other. Every protocol and message must have a key: for brevity, the key of a message equals the protocol key parameters that feature in it. An enactment is *complete* when all its public ⌜in⌝ and ⌜out⌝ parameters are bound. Specifically, an enactment of *Purchase* must create a tuple of bindings for its four public parameters but may omit address and response, which are private.

Listing 1 involves a private parameter response that is ⌜out⌝ in both *accept* and *reject*. Since BSPL models each enactment as producing a tuple of parameter bindings, the existence of the same parameter with an ⌜out⌝ adornment in two messages indicates an integrity violation: thus the two messages cannot both occur: if they did the binding would be conceptually undefined. That is, *accept* and *reject* conflict on response. And, outcome is ⌜out⌝ in both *reject* and *deliver*, thereby causing a conflict between them.

Syntax

The following BSPL syntax and explanations are simplified from Singh [13]. Superscripts of $+$ and $*$ indicate one or more and zero or more repetitions, respectively. Below, ⌊ and ⌋ delimit expressions, considered optional if without a superscript. For simplicity, we omit cardinality restrictions and parameter types.

L_1. A protocol declaration consists of a name, two or more roles, one or more parameters, and one or more references to constituent protocols or messages. The parameters marked key together form this declaration's key.
Protocol ⟶ *Name* { role *Role*$^+$ parameter ⌊*Parameter*⌊key⌋⌋$^+$*Reference** }

L_2. A reference to a protocol (from a declaration) consists of the name of the protocol appended by as many roles and parameters as it declares. At least one parameter of the reference must be a key parameter of the declaration in which it occurs.
Reference ⟶ *Name* (*Role*$^+$ *Parameter*$^+$)

L_3. Alternatively, a reference is a message schema, and consists of exactly one name, exactly two roles, and one or more parameters (at least one of which must be a key parameter).
Reference ⟶ *Role* ↦ *Role* : *Name* [*Parameter*$^+$]

L_4. Each parameter consists of an adornment and a name.
Parameter ⟶ *Adornment Name*

L_5. An adornment is usually either ⌜in⌝ or ⌜out⌝. A ⌜nil⌝ in a reference indicates an unknown parameter.
Adornment ⟶ in | out | nil

3. INTUITIONS ON BSPL SEMANTICS

A protocol describes an interaction by specifying messages to be exchanged between specific roles, and by (indirectly, though effectively) imposing a partial order on the messages. An enactment of a protocol involves each of its roles being adopted by an agent, and the agents exchanging messages that the protocol specifies. A message instantiates a message schema and is precisely described by its name, its sender, receiver, and bindings for each of its parameters.

BSPL is characterized by the interplay between parameters and messages. In describing interactions declaratively, we are concerned with tuples of parameter bindings. The keys determine the units of enactment. And, parameter bindings are immutable in any enactment. The parameter adornments determine how information is propagated through them. Interactions are realized exclusively through the exchange of messages: everything of relevance to the interaction is visible in a message emission and reception.

For example, *quote* (for a given ID, its key) may occur only after ID and item are bound. An enactment of a protocol may begin only when at least one of its messages is enabled. However, an enactment begins and proceeds to completion only if the agents involved decide autonomously to do their respective parts.

3.1 Knowledge and Viability

We distinguish between an agent's local and internal states. The *local* state is public, though limited to the role's view of the protocol enactment. The *internal* state depends on the agent implementation and is not visible to any other agent. We consider only the local states of roles. The history of a role maps naturally to its local state: each message emitted or received advances the local state.

Figure 1 illustrates the impact on a sender and receiver's knowledge with respect to a parameter adorned ⌜in⌝, ⌜out⌝, or ⌜nil⌝. Further, because of the immutability of parameter bindings, the knowledge of a role increases monotonically as its history extends. Thus these are the only three possible adornments of a parameter.

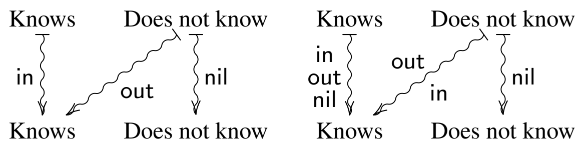


Figure 1: Viability and knowledge effects for an adorned parameter on the sender (left) and receiver (right) of a message.

As a result, an ⌜in⌝ emission must be preceded by an ⌜in⌝ or an ⌜out⌝ emission or reception and an ⌜out⌝ or a ⌜nil⌝ emission must not be preceded by an ⌜in⌝ or an ⌜out⌝ emission or reception. Notice that to send a message with an ⌜out⌝ parameter, the sender agent computes the parameter (through its internal business logic) but the only role states, i.e., local states, in which the message can be emitted are those where the role does not know already what it is. A correctly implemented agent would not compute a binding for an ⌜out⌝ parameter that (for the given keys) is already known to its role. And, for an ⌜in⌝ parameter, the role must know its binding through a previous message. Such parameter-based causality constraints underlie the semantics of BSPL.

Figure 1 identifies all the *viable* message emissions and receptions given a role's state of knowledge with respect to a parameter in a message. A message reception is always viable. In a practical system, we would validate incoming messages, as the LoST middleware [14] does, but in the abstract semantics we assume that the local state is never corrupted. For a message emission, the sender must already know the bindings of the ⌜in⌝ parameters and *not* know the bindings for any of the other parameters.

3.2 Causal Structures

Because BSPL incorporates a flexible description of interactions, it matches well with our computational approach of generating a *causal structure* as a set of declarative (temporal) constraints that capture the flow of causality within and across roles. A causal structure identifies partial states for each role that describe some parameters as known, some as unknown, and leave others indeterminate. It supports reasoning to verify various properties.

BSPL's flexibility does not accord well with detailed graphical representations of possible enactments, which get unwieldy fast. Specifically, a causal structure may be mapped to a finite state machine but with an explosion in states and transitions. However, to convey some intuitions, we show some *informal* pictures below.

Consider protocol *Sequential*. Because ID is ⌜out⌝ in *initial* and ⌜in⌝ in *additional*, *additional* causally depends on *initial*.

```
Sequential { ... // Omitting roles
 parameter out ID key, out answer, out more
 B ↦ S: initial [out ID, out answer]
 B ↦ S: additional [in ID, out more]
}
```

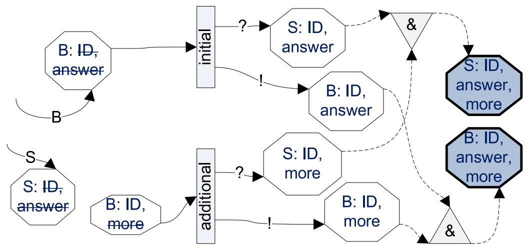
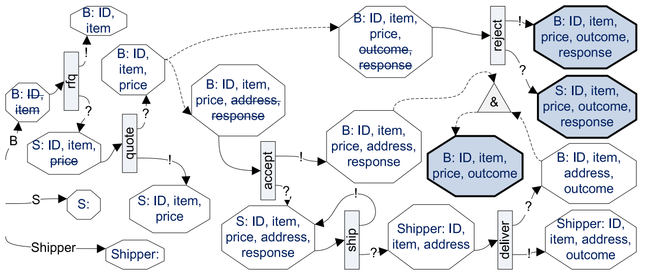


Figure 2: *Sequential* is enactable and safe.

Sequential's causal structure (Figure 2) represents the partial states of each role. It shows each message along with a precondition partial state in which it might be emitted and the effects it would have on the states of its sender and receiver. The precondition specifies what parameters its sender must know (⌜in⌝ parameters) and must not know (⌜out⌝ and ⌜nil⌝ parameters) before emitting the message. The effects specify what parameters its sender and receiver must know after it occurs (⌜out⌝ and ⌜in⌝ parameters). The solid transitions capture the intuitions of Figure 1. (The dashed lines show logical relationships.) A message can be emitted in any state that matches its precondition: the sender should know *all* parameters written plain and know *none* of the parameters written with a strikethrough line.

In *Sequential*, role B can send only *initial* at the outset. Upon emitting and receiving this message, respectively, B and S's states change as specified by the ? and ! edges from the message node. Thus, B knows answer and so cannot send *initial* but it would be superfluous here anyway. Also, B knows ID and can send *additional*, resulting in changing B and S's states further. When both messages are emitted and received, each ⌜out⌝ parameter of *Sequential* is known to at least one role, i.e., the enactment completes.

Figure 3 shows the causal structure for *Purchase*. The buyer B emits an *rfq*, which enables the seller S to send a *quote*. At this point, B has a choice about whether to *accept* or *reject*. In case of *reject*, all public parameters are bound so the enactment completes. In case of *accept*, S may send *ship* to the SHIPPER, who can DELIVER the item to B, thereby completing the enactment.



B: ID, item
!
rfq
B: ~~ID,~~ ~~item~~
?
B
S: ID, item, ~~price~~
B: ID, item, price
?
quote
!
S
S:
S: ID, item, price
Shipper
Shipper:
B: ID, item, price, ~~address,~~ ~~response~~
accept
!
?
B: ID, item, price, address, response
S: ID, item, price, address, response
ship
!
?
Shipper: ID, item, address
deliver
?
!
B: ID, item, price, ~~outcome,~~ ~~response~~
reject
!
?
B: ID, item, price, outcome, response
S: ID, item, price, outcome, response
&
B: ID, item, price, outcome
B: ID, item, address, outcome
Shipper: ID, item, address, outcome

Figure 3: Causal structure for *Purchase*.

3.3 Enactability

The intuition behind enactability simply is that a protocol should provide a clear path to completion, i.e., generating a tuple for all ⌜in⌝ and ⌜out⌝ public parameters. *Sequential* and *Purchase* are enactable, as the enactments described above demonstrate. Let's consider *Local Conflict*, whose causal structure is in Figure 4.

```
Local Conflict { ...
 parameter out ID key, out answer, out alternative
 B ↦ C: one [out ID, out answer]
 B ↦ C: two [out ID, out alternative]
}
```

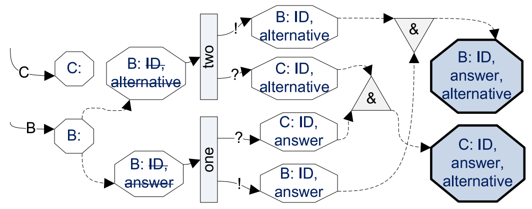


Figure 4: *Local Conflict* is not enactable, but is safe.

Local Conflict is not enactable. Because messages *one* and *two* conflict on ID, emitting *one* disables *two* and vice versa. Thus, at most one of them can be emitted for a given binding of ID. Hence, either answer or alternative would necessarily remain unbound.

3.4 Safety

A useful protocol must be safe, meaning that each parameter must have no more than one binding for the same key in any enactment. *Sequential* is safe because each of its parameters is ⌜out⌝ in at most one message.

Safety requires that at most one message instance with the same parameter adorned ⌜out⌝ occurs for any key. This condition is easy to ensure for a single role. Specifically, when a role emits a message with an ⌜out⌝ parameter, its knowledge changes and it may send no subsequent messages with the same ⌜out⌝ parameter. The BSPL semantics requires this local constraint and the LoST middleware [14] enforces it. In essence, the agent should decide internally which, if any, of the conflicting messages to send. *Local Conflict* is safe because B is the sender of both conflicting messages.

But no such reasoning applies *across* senders because each maintains separate local state information. For example, *Abrupt Cancel* below is not safe because it involves a race between B and S. Consider *Abrupt Cancel* and its causal structure (Figure 5).

```
Abrupt Cancel { ...
 B ↦ S: order [out ID, out item]
 B ↦ S: cancel [in ID, in item, out outcome]
 S ↦ B: goods [in ID, in item, out outcome]
}
```

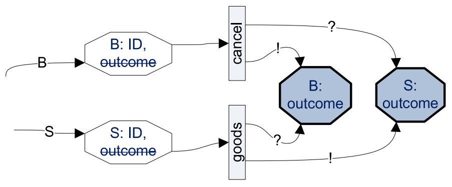


Figure 5: *Abrupt Cancel* is enactable but not safe.

Abrupt Cancel is enactable because it is possible to go from its initial states to where all its public parameters bound. However, because messages *goods* and *cancel* may both be emitted, outcome may be bound twice. Thus *Abrupt Cancel* is not safe.

To verify safety involves checking that the protocol prevents a situation where two roles can both be enabled to send conflicting messages. In essence, if there are two conflicting execution paths, they must have a prior branching point controlled by the same role. For example, in Listing 1, the conflict between *accept* and *reject* on private parameter response means that at most one of these two messages may occur. The same sender, B, is involved, so *Purchase* is safe. Further, because *deliver* can only occur if *ship* occurs previously (to convey address) and *ship* can occur only if *accept* occurs prior (to produce a binding for address). Thus *deliver* presupposes *accept* but *accept* and *reject* conflict. Therefore, the mutual exclusion between *deliver* and *reject* is guaranteed.

Listing 2 shows *Purchase Unsafe* based on *Purchase*, and which dispenses with the private parameter response. Thus *accept* and *reject* no longer conflict, and B may send both of them. Thus *deliver* and *reject* may both occur, violating safety for outcome.

Listing 2: An unsafe variant of *Purchase*.

```
Purchase Unsafe{ // Same as Purchase except these
 B ↦ S: accept[in ID, in item, in price, out
      address]
 B ↦ S: reject[in ID, in item, in price, out
      outcome]
}
```

In *Abrupt Cancel*, the inconsistency is obvious because different roles make mutually inconsistent decisions. *Purchase Unsafe* is more insidious because the inconsistent decisions by B and SHIPPER are gated by a decision by B. Our intuition may indicate that B acts in an odd manner when it emits both *accept* and *reject*. However, the semantics of a protocol depends *only* on the protocol specification, not on any imagined internal policies of the agents adopting its various roles. The protocol specification is the only means by which we constrain such policies: there is no hidden additional specification. (Of course, an autonomous agent may violate any protocol but the semantics tells us clearly what is a violation. Notice that LoST [14] helps an agent both respect a given protocol itself and ensure that others are respecting the protocol too.)

In general, when different roles are the senders of two conflicting messages, consistency is enforceable only if there is a causally prior conflict produced by the same role. Specifically, safety holds precisely when no causal path on which an ⌜out⌝ parameter is bound once may have the same ⌜out⌝ parameter bound again.

3.5 Liveness

Consider the following variant of *Purchase*.

```
Purchase No Ship { ...
//Same as Purchase but with ship deleted
}
```

Purchase No Ship remains enactable because if B emits *reject*, all its public parameters are bound. Despite this, however, if B emits *accept* the protocol enactment cannot complete: outcome is never bound because B can no longer send *reject* and the SHIPPER never becomes enabled to send *deliver*. Notice that we can never require that an agent send any message. And, in some cases, the protocol semantics prevents an agent from legally emitting a message.

Liveness requires that no matter what messages any of the agents has emitted, it should always be possible for a protocol enactment to legally complete. Liveness does not mean that the completion is necessarily a "happy" one from the application standpoint, just that the enactment terminates. In this sense, *Purchase* is live as are *Purchase Unsafe* and *Abrupt Cancel*. *Local Conflict* is not live.

Liveness entails enactability but not the other way around. Liveness is the more fundamental property. However, during design, enactability can help catch errors that are easier to fix, whereas to make corrections to ensure liveness can be more demanding.

4. FORMALIZING BSPL SEMANTICS

We now formalize the above intuitions. For convenience, we fix the symbols by which we refer to finite lists (mostly, treated as sets) of roles ($\vec{t}$), public roles ($\vec{x}$), private roles ($\vec{y}$), public parameters ($\vec{p}$), key parameters ($\vec{k} \subseteq \vec{p}$), ⌜in⌝ parameters ($\vec{p_I} \subseteq \vec{p}$), ⌜out⌝ parameters ($\vec{p_O} \subseteq \vec{p}$), ⌜nil⌝ parameters ($\vec{p_N} \subseteq \vec{p}$), private parameters ($\vec{q}$), and parameter bindings ($\vec{v}, \vec{w}$). Here, $\vec{p} = \vec{p_I} \cup \vec{p_O} \cup \vec{p_N}$, $\vec{p_I} \cap \vec{p_O} = \emptyset$,

$\vec{p_I} \cap \vec{p_N} = \emptyset$, and $\vec{p_N} \cap \vec{p_O} = \emptyset$. Also, t and p refer to an individual role and parameter, respectively. To reduce notation, we rename private roles and parameters to be distinct in each protocol, and the public roles and parameters of a reference to match the declaration in which they occur. Throughout, we use $\downarrow_x$ to project a list to those of its elements that belong to x.

Definition 1 captures BSPL protocols. A protocol (via any of its parameters) may reference another protocol (via its public parameters). The references bottom out at message schemas. Above, *Purchase* references *accept*. And, if a protocol were to reference *Purchase*, it would be able to reference (from its public or private parameters) only the public parameters of *Purchase*, not address.

DEFINITION 1. A *protocol* $\mathscr{P}$ is a tuple $\langle n, \vec{x}, \vec{y}, \vec{p}, \vec{k}, \vec{q}, F \rangle$, where n is a name; $\vec{x}$, $\vec{y}$, $\vec{p}$, $\vec{k}$, and $\vec{q}$ are as above; and F is a finite set of f references, $\{F_1, \ldots, F_f\}$. $(\forall i : 1 \leq i \leq f \Rightarrow F_i = \langle n_i, \vec{x_i}, \vec{p_i}, \vec{k_i} \rangle$, where $\vec{x_i} \subseteq \vec{x} \cup \vec{y}$, $\vec{p_i} \subseteq \vec{p} \cup \vec{q})$, $\vec{k_i} = \vec{p_i} \cap \vec{k}$, and $\langle n_i, \vec{x_i}, \vec{p_i}, \vec{k_i} \rangle$ is the public projection of a protocol $\mathscr{P}_i$ (with roles and parameters renamed).

DEFINITION 2. The *public projection* of a protocol $\mathscr{P} = \langle n, \vec{x}, \vec{y}, \vec{p}, \vec{k}, \vec{q}, F \rangle$ is given by the tuple $\langle n, \vec{x}, \vec{p}, \vec{k} \rangle$.

We treat a message schema $\ulcorner s \mapsto r : m\ \vec{p}(\vec{k}) \urcorner$ as an atomic protocol with exactly two roles (sender and receiver) and no references: $\langle m, \{s, r\}, \emptyset, \vec{p}, \vec{k}, \emptyset, \emptyset \rangle$. Here $\vec{k}$ is the set of key parameters of the message schema. Usually, $\vec{k}$ is understood from the protocol in which the schema is referenced: $\vec{k}$ equals the intersection of $\vec{p}$ with the key parameters of the protocol declaration.

Below, let $\text{roles}(\mathscr{P}) = \vec{x} \cup \vec{y} \cup \bigcup_i \text{roles}(F_i)$; $\text{params}(\mathscr{P}) = \vec{p} \cup \vec{q} \cup \bigcup_i \text{params}(F_i)$; $\text{msgs}(\mathscr{P}) = \bigcup_i \text{msgs}(F_i)$ and $\text{msgs}(s \mapsto r : m\ \vec{p}) = \{m\}$. Definition 3 assumes that the message instances are unique up to the key specified in their schema.

DEFINITION 3. A *message instance* $m[s, r, \vec{p}, \vec{v}]$ associates a message schema $\ulcorner s \mapsto r : m\ \vec{p}(\vec{k}) \urcorner$ with a list of values, where $|\vec{v}| = |\vec{p}|$, where $\vec{v} \downarrow_p = \ulcorner \text{nil} \urcorner$ iff $p \in \vec{p_N}$.

Definition 4 introduces a *universe of discourse (UoD)*. Definition 5 captures the idea of a *history* of a role as a sequence (equivalent to a set in our approach) of all and only the messages the role either emits or receives. Thus H^ρ captures the local view of an agent who might adopt role ρ during the enactment of a protocol. A history may be infinite in general but we assume each enactment in which a tuple of parameter bindings is generated is finite.

DEFINITION 4. A *UoD* is a pair $\langle \mathcal{R}, \mathcal{M} \rangle$, where $\mathcal{R}$ is a set of roles, $\mathcal{M}$ is a set of message names; each message specifies its parameters along with its sender and receiver from $\mathcal{R}$.

DEFINITION 5. A *history* of a role ρ, H^ρ, is given by a sequence of zero or more message instances $m_1 \circ m_2 \circ \ldots$. Each m_i is of the form $m[s, r, \vec{p}, \vec{v}]$ where $\rho = s$ or $\rho = r$, and $\circ$ means sequencing.

Definition 6 captures the idea that what a role knows at a history is exactly given by what the role has seen so far in terms of incoming and outgoing messages. Here, 2(i) ensures that $m[s, r, \vec{p}(\vec{k}), \vec{v}]$, the message under consideration, does not violate the uniqueness of the bindings. And, 2(ii) ensures that ρ knows the binding for each $\ulcorner \text{in} \urcorner$ parameter and not for any $\ulcorner \text{out} \urcorner$ or $\ulcorner \text{nil} \urcorner$ parameter.

DEFINITION 6. A message instance $m[s, r, \vec{p}(\vec{k}), \vec{v}]$ is *viable* at role ρ's history H^ρ iff (1) $r = \rho$ (reception) or (2) $s = \rho$ (emission) and (i) $(\forall m_i[s_i, r_i, \vec{p_i}, \vec{v_i}] \in H^\rho$ if $\vec{k} \subseteq \vec{p_i}$ and $\vec{v_i} \downarrow_{\vec{k}} = \vec{v} \downarrow_{\vec{k}}$ then $\vec{v_i} \downarrow_{\vec{p} \cap \vec{p_i}} = \vec{v} \downarrow_{\vec{p} \cap \vec{p_i}})$ and (ii) $(\forall p \in \vec{p} : p \in \vec{p_I}$ iff $(\exists m_i[s_i, r_i, \vec{p_i}, \vec{v_i}] \in H^\rho$ and $p \in \vec{p_i}$ and $\vec{k} \subseteq \vec{p_i}))$.

Definition 7 captures that a *history vector* for a protocol is a vector of histories of role that together are causally sound: a message is received only if it has been emitted [8].

DEFINITION 7. Let $\langle \mathcal{R}, \mathcal{M} \rangle$ be a UoD. We define a *history vector* for $\langle \mathcal{R}, \mathcal{M} \rangle$ as a vector $[H^1, \ldots, H^{|\mathcal{R}|}]$, such that $(\forall s, r : 1 \leq s, r \leq |\mathcal{R}| : H^s$ is a history and $(\forall m[s, r, \vec{p}, \vec{v}] \in H^r : m \in \mathcal{M}$ and $m[s, r, \vec{p}, \vec{v}] \in H^s))$.

The progression of a history vector records the progression of an enactment of a multiagent system. Under the above causality restriction, a vector that includes a reception must have progressed from a vector that includes the corresponding emission. Further, we make no FIFO assumption about message delivery. The viability of the messages emitted by any role ensures that the progression is epistemically correct with respect to each role.

DEFINITION 8. A history vector over $\langle \mathcal{R}, \mathcal{M} \rangle$, $[H^1, \ldots, H^{|\mathcal{R}|}]$, is *viable* iff either (1) each of its element histories is empty or (2) it arises from the progression of a viable history vector through the emission or the reception of a viable message by one of the roles, i.e., $(\exists i, m_j : H^i = H'^i \circ m_j$ and $[H^1, \ldots, H'^i, H^{|\mathcal{R}|}]$ is viable).

The heart of our formal semantics is the *intension* of a protocol, defined relative to a UoD, and given by the set of viable history vectors, each corresponding to its successful enactment. Given a UoD, Definition 9 specifies a universe of enactments, based on which we express the intension of a protocol. We restrict attention to viable vectors because those are the only ones that can be realized. We include private roles and parameters in the intension so that compositionality works out. In the last stage of the semantics, we project the intension to the public roles and parameters.

DEFINITION 9. Given a UoD $\langle \mathcal{R}, \mathcal{M} \rangle$, the *universe of enactments* for that UoD, $\mathcal{U}_{\mathcal{R},\mathcal{M}}$, is the set of viable history vectors, each of which has exactly $|\mathcal{R}|$ dimensions and each of whose messages instantiates a schema in $\mathcal{M}$.

Definition 10 states that the intension of a message schema is given by the set of viable history vectors on which that schema is instantiated, i.e., an appropriate message instance occurs in the histories of both its sender and its receiver.

DEFINITION 10. The intension of a message schema is given by: $[\![m(s, r, \vec{p})]\!]_{\mathcal{R},\mathcal{M}} = \{H | H \in \mathcal{U}_{\mathcal{R},\mathcal{M}}$ and $(\exists \vec{v}, i, j : H^s_i = m[s, r, \vec{p}, \vec{v}]$ and $H^r_j = m[s, r, \vec{p}, \vec{v}])\}$.

A (composite) protocol completes if one or more of subsets of its references completes. For example, *Purchase* yields two such subsets, namely, $\{$*rfq*, *quote*, *accept*, *ship*, *deliver*$\}$ and $\{$*rfq*, *quote*, *reject*$\}$. Informally, each such subset contributes all the viable interleavings of the enactments of its members, i.e., the intersection of their intensions. Definition 11 captures the *cover* as an adequate subset of references of a protocol, and states that the intension of a protocol equals the union of the contributions of each of its covers.

DEFINITION 11. Let $\mathscr{P} = \langle n, \vec{x}, \vec{y}, \vec{p}, \vec{k}, \vec{q}, F \rangle$ be a protocol. Let cover $(\mathscr{P}, G) \equiv G \subseteq F | (\forall p \in \vec{p} : (\exists G_i \in G : G_i = \langle n_i, x_i, p_i \rangle$ and $p \in \vec{p_i}))$; and $\mathscr{P}$'s intension, $[\![\mathscr{P}]\!]_{\mathcal{R},\mathcal{M}} = (\bigcup_{\text{cover}(\mathscr{P},G)} (\bigcap_{G_i \in G} [\![G_i]\!]_{\mathcal{R},\mathcal{M}})) \downarrow_{\vec{x}}$.

As an example, consider a message m_1 with a single key parameter p adorned $\ulcorner \text{in} \urcorner$ whose sender is role r_1. The intension of this message with respect to a UoD $\langle \mathcal{R}, \mathcal{M} \rangle$ can be nonempty only if one of the following conditions holds for some $m_2 \in \mathcal{M}$ (m_2 should precede m_1 in role r_1's history):

- m_2's schema involves the parameter p adorned $\ulcorner \text{out} \urcorner$ and r_1 is the sender or receiver of m_2.
- m_2's schema involves the parameter p adorned $\ulcorner \text{in} \urcorner$ and the receiver of m_2 is the same role r_1.

The intension of m_1 would still be empty if the intension of any such m_2 is empty, e.g., if m_2 does not occur on any viable history vector. In general, if message m_1 has message m_2 as an essential prerequisite, then the intension of m_1 must be a subset of the intension of m_2, i.e., $[\![m_1]\!] \subseteq [\![m_2]\!]$.

The UoD of protocol $\mathscr{P}$ consists of $\mathscr{P}$'s roles and messages including its references recursively. For example, *Purchase*'s UoD $U = \langle\{\text{B, S, SHIPPER}\}, \{\mathit{rfq}, \mathit{quote}, \mathit{accept}, \mathit{reject}, \mathit{ship}, \mathit{deliver}\}\rangle$.

DEFINITION 12. The *UoD of a protocol* $\mathscr{P}$, $\text{UoD}(\mathscr{P}) = \langle \text{roles}(\mathscr{P}), \text{msgs}(\mathscr{P})\rangle$.

4.1 Enactability, Safety, and Liveness

Enactability means that we can produce a history vector that generates bindings for all public parameters. Note that a protocol that has a public ⌜in⌝ parameter is not enactable standalone, and must be referenced from another protocol. Safety means that we cannot produce a history vector that generates more than one binding for any parameter. Liveness means that we cannot produce a history vector that deadlocks.

DEFINITION 13. A protocol $\mathscr{P}$ is *enactable* iff $[\![\mathscr{P}]\!]_{\text{UoD}(\mathscr{P})} \neq \emptyset$.

DEFINITION 14. A protocol $\mathscr{P}$ is *safe* iff each history vector in $[\![\mathscr{P}]\!]_{\text{UoD}(\mathscr{P})}$ is safe. A history vector is *safe* iff all key uniqueness constraints apply across all histories in the vector.

DEFINITION 15. A protocol $\mathscr{P}$ is *live* iff each history vector in the universe of enactments $\text{UoD}(\mathscr{P})$ can be extended through a finite number of message emissions and receptions to a history vector in $\text{UoD}(\mathscr{P})$ that is complete.

4.2 Well-Formedness Conditions

Because BSPL relies upon parameter adornments and keys for causality and integrity, it is essential that a protocol meet some elementary syntactic conditions. The main idea is that each key (set of parameters) identifies a logical entity and nonkey parameters express attributes of that entity. Thus the nonkey parameters have no meaning if separated from their keys. As an example, think of taking the age of a person with an identifier and putting the age in a record without the person's identifier. We need to carry the original identifier along. Of course, an agent may copy the contents of one parameter to another (e.g., set price equal to age), but such internal reasoning is not in the purview of BSPL.

The foregoing motivation leads to the following constraints. First, two messages must involve different parameters unless the key of one is a subset of the key of the other. Second, no message m_0 that has an ⌜out⌝ key parameter must have an ⌜in⌝ nonkey parameter p unless: if p occurs as an ⌜out⌝ in a message m_1 then m_0 includes m_1's key; and, if p occurs as an ⌜in⌝ in a message m_2 with an entirely ⌜in⌝ key then m_0 includes m_2's key. Notice that Definition 6 seeks to respect the above constraint. The net result is that if a message has an ⌜out⌝ parameter p then the key with which p is produced must be stated whenever we use p.

5. VERIFYING BSPL PROTOCOLS

To verify BSPL protocols, we express a causal structure as well as the target properties in a temporal language and determine the satisfiability of the resulting expressions. The language we adopt, Precedence, is an extension of Singh's [12] language.

The atoms of Precedence are events. Below, e and f are events. If e is an event, its complement $\overline{e}$ is also an event. Precedence treats e and $\overline{e}$ on par. The terms $e \cdot f$ and $e \star f$, respectively, mean that e occurs prior to f and e and f occur simultaneously. The Boolean operators: '$\vee$' and '$\wedge$' have the usual meanings. The syntax follows conjunctive normal form:

L$_6$. $I \longrightarrow \mathit{clause} \mid \mathit{clause} \wedge I$

L$_7$. $\mathit{clause} \longrightarrow \mathit{term} \mid \mathit{term} \vee \mathit{clause}$

L$_8$. $\mathit{term} \longrightarrow \mathit{event} \mid \mathit{event} \cdot \mathit{event} \mid \mathit{event} \star \mathit{event}$

The semantics of Precedence is given by pseudolinear runs of events (instances): "pseudo" because several events may occur together though there is no branching. Let Γ be a set of events where $e \in \Gamma$ iff $\overline{e} \in \Gamma$. A run is a function from natural numbers to the power set of Γ, i.e., $\tau : \mathbb{N} \mapsto 2^{\Gamma}$. The i^{th} index of τ, $\tau_i = \tau(i)$. The length of τ is the first index i at which $\tau(i) = \emptyset$ (after which all indices are empty sets). We say τ is empty if $|\tau = 0$. The subrun from i to j of τ is notated $\tau_{[i,j]}$. Its first $j - i + 1$ values are extracted from τ and the rest are empty, i.e., $\tau_{[i,j]} = \langle \tau_i, \tau_{i+1} \ldots \tau_{j-i+1} \ldots \emptyset \ldots\rangle$. On any run, e or $\overline{e}$ may not both occur. Events are nonrepeating.

$\tau \models_i E$ means that τ satisfies E at i or later. We say τ is a *model* of expression E iff $\tau \models_0 E$. E is *satisfiable* iff it has a model.

M$_1$. $\tau \models_i e$ iff $(\exists j \geq i : e \in \tau_j)$

M$_2$. $\tau \models_i e \star f$ iff $(\exists j \geq i : \{e, f\} \subseteq \tau_j)$

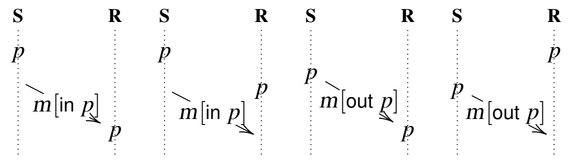
M$_3$. $\tau \models_i r \vee u$ iff $\tau \models_i r$ or $\tau \models_i u$

M$_4$. $\tau \models_i r \wedge u$ iff $\tau \models_i r$ and $\tau \models_i u$

M$_5$. $\tau \models_i e \cdot f$ iff $(\exists j \geq i : \tau_{[i,j]} \models_0 e$ and $\tau_{[j+1,|\tau|]} \models_0 f)$

We capture the local state of each role in a BSPL protocol by defining two kinds of events: (1) a particular message having been observed (one event each for sender and receiver) and (2) a particular parameter having a known binding (one event for each parameter whether emitted or received in any message).

Although our approach is generic and implemented, we describe it via examples based on *Purchase* to simplify the exposition in limited space. From *Purchase*, we first determine events from messages (B: quote, S: quote, ...: total 12) and parameters (B: price, S: price, ...: total 17). Next, we describe how we generate a causal structure, ignoring ⌜nil⌝ adornments for brevity. For protocol $\mathscr{P}$, let $\mathscr{C}_P$ be the conjunction of all clauses of the following types.



(a) ⌜in⌝ parameters (b) ⌜out⌝ parameters

Figure 6: Treatment of ⌜in⌝ and ⌜out⌝ parameters: each has one possible scenario for the sender and two for the receiver.

Reception. If a message is received, it was previously emitted. Specifically, either the receiver role never observes a message or the sender observes it before (six clauses).

$\overline{\text{B: quote}} \vee \text{S: quote} \cdot \text{B: quote}$

Information transmission. For each message, in its sender's view.

See Figure 6(a). Either the message is never emitted or each of its ⌜in⌝ parameters is observed before (14 clauses).

$\overline{\text{S: quote}} \vee \text{S: item} \cdot \text{S: quote}$

See Figure 6(b). Either the message is never emitted or it is observed simultaneously with each of its ⌜out⌝ parameters (eight clauses).

$\overline{\text{S: quote}} \vee \text{S: price} \star \text{S: quote}$

Information reception. For each message, in its receiver's view.

Any ⌜out⌝ or ⌜in⌝ parameter occurs before or simultaneously with the message. In other words, either the message is not observed or each such parameter is observed no later than the message. A parameter may be observed earlier through some other path [14] (22 clauses). See Figure 6.

$\overline{\text{B: quote}} \vee \text{B: price} \cdot \text{B: quote} \vee \text{B: price} \star \text{B: quote}$

Information minimality. For any role, if a parameter occurs, it occurs simultaneously with *some* message emitted or received. Thus, no role observes a parameter noncausally (16 clauses).

$\overline{\mathsf{B\colon price}} \vee \mathsf{B\colon price} \star \mathsf{B\colon quote} \vee \mathsf{B\colon price} \star \mathsf{B\colon accept} \vee \mathsf{B\colon price} \star \mathsf{B\colon reject}$

Ordering. If a role emits any two messages, it observes them in some order, not simultaneously. This constraint rules out lockstep enactments whereby two messages happen to be emitted magically at the same time to escape our causality constraints (four clauses).

$\overline{\mathsf{S\colon quote}} \vee \overline{\mathsf{S\colon ship}} \vee \mathsf{S\colon quote} \cdot \mathsf{S\colon ship} \vee \mathsf{S\colon ship} \cdot \mathsf{S\colon quote}$

The correctness of our decision procedures relies on the following result, whose proof follows from construction.

THEOREM 1. Given a well-formed protocol $\mathscr{P}$, for every viable history vector, there is a model of $\mathscr{C}_P$ and vice versa.

Proof Sketch. In the forward direction, we can proceed by induction to find a pseudolinear run that includes each message emission and reception that occurs in any history in the history vector H. An empty history vector corresponds to an empty run. Inductively assume that a run τ exists for history vector H. We can extend H via the emission of a message only if the message is viable in H, meaning that its sender has locally observed its $\ulcorner$in$\urcorner$ parameters but not its $\ulcorner$out$\urcorner$ or $\ulcorner$nil$\urcorner$ parameters. Using the information transmission clauses, we can construct a run τ' that extends τ with the message emission event. We can extend H via the reception of the message by a role r. By Definition 7, H must include a message emission event for some role sending to r. Thus we can construct τ' as τ appended with the message reception. In the reverse direction, given a run τ, we simply construct each member history of the vector from τ by appending the message emission and reception events (and ignoring all others) involving a role in sequence to that role's history. The clauses in $\mathscr{C}_P$ ensure that the resulting vector is viable. □

5.1 Verifying Enactability

We generate clauses that together indicate completion of a protocol enactment. For each public parameter we identify all the messages in which it occurs and specify a clause that is the disjunction of the receivers of those messages observing that parameter. For example, for outcome, we have $\mathsf{B\colon outcome} \vee \mathsf{S\colon outcome}$ because B and S are the receivers of the two messages in which outcome occurs. For protocol $\mathscr{P}$, let $\mathscr{E}_P$ be the conjunction of all such clauses. Our decision procedure is simply to check if $\mathscr{C}_P \wedge \mathscr{E}_P$ is satisfiable.

THEOREM 2. A well-formed protocol $\mathscr{P}$ is *enactable* if and only if $\mathscr{C}_P \wedge \mathscr{E}_P$ is satisfiable.

Proof Sketch. In the forward direction, assume protocol $\mathscr{P}$ has a nonempty intension. Then, by Definition 11, it has a cover of references that yield bindings for all its public parameters, indicating that $\mathscr{E}_P$ holds for each vector. Further from Theorem 1, we know that $\mathscr{C}_P$ holds in the corresponding run. In the reverse direction, from any run that satisfies $\mathscr{C}_P \wedge \mathscr{E}_P$ we can identify a cover with a nonempty intension. □

5.2 Verifying Safety

Safety means that for each parameter (public or private) adorned $\ulcorner$out$\urcorner$ in two or more messages, no more than one of those "competing" messages may be emitted. To this end, we generate clauses expressing that two or more of the competing messages of some parameter are observed by their sender. For *Purchase*, outcome and response are the relevant parameters. Therefore, the resulting two clauses are $(\mathsf{B\colon accept} \vee \mathsf{SHIPPER\colon deliver})$ and $(\mathsf{B\colon reject})$. This clause says that both reject and either accept or deliver is emitted, which signifies an inconsistency. For protocol $\mathscr{P}$, let $\mathscr{S}_P$ be the conjunction of all the property clauses. Our decision procedure is simply to check if $\mathscr{C}_P \wedge \mathscr{S}_P$ is unsatisfiable.

THEOREM 3. A well-formed protocol $\mathscr{P}$ is *safe* if and only if $\mathscr{C}_P \wedge \mathscr{S}_P$ is *not* satisfiable.

Proof Sketch. Let τ satisfy $\mathscr{C}_P \wedge \mathscr{S}_P$. Then, by Theorem 1, we can construct a history vector in which at least two conflicting messages occur. We can extend such a vector to one where at least two roles generate bindings for the same parameter, thus violating integrity (we cannot prove that they will generate the same bindings since we have no access to internal reasoning). Conversely, if $\mathscr{C}_P \wedge \mathscr{S}_P$ is *not* satisfiable, there is no history vector in the intension of $\mathscr{P}$ that violates integrity. □

5.3 Verifying Liveness

Notice that a specific protocol enactment being incomplete does not entail that some role is blocked. A enactment may fail to complete even though the protocol may be live: (1) one or more agents may decide not to send messages or (2) one or more messages may be lost—causality requires only that receptions are preceded by emissions, not that emissions are always followed by receptions.

To avoid situations where some agents may decide not to send any messages, we restrict attention to models that are *maximal* in the sense that they have no message left unemitted that could be emitted based on the parameters that feature in it. That is, in a maximal model, if the sender has observed the $\ulcorner$in$\urcorner$ and not observed the $\ulcorner$out$\urcorner$ parameters of a message, then the sender must also observe the message. The following says that either S emits *quote* or it does not observe ID or item or it observes price, i.e., $(\mathsf{S\colon quote} \vee \overline{\mathsf{S\colon ID}} \vee \overline{\mathsf{S\colon item}} \vee \mathsf{S\colon price})$.

To avoid situations where the communication infrastructure may drop messages, we constrain our model to those where every message emitted is delivered, e.g., $(\overline{\mathsf{S\colon quote}} \vee \mathsf{B\colon quote})$. Third, the enactment is incomplete, which means that at least one of the public $\ulcorner$out$\urcorner$ parameters remains unbound at each role. The above condition yields a clause $(\overline{\mathsf{:ID}} \vee \overline{\mathsf{:item}} \vee \overline{\mathsf{:price}} \vee \overline{\mathsf{:outcome}})$ constructed from protocol-level literals, for which no role is relevant. For each such literal, we assert two clauses $(\mathsf{:price} \vee \overline{\mathsf{B\colon price}} \vee \overline{\mathsf{S\colon price}})$ and $(\overline{\mathsf{:price}} \vee \mathsf{B\colon price} \vee \mathsf{S\colon price})$, meaning that the protocol-level literal is true exactly if at least one role observes the parameter.

If a maximal, nonlossy enactment can be incomplete that means the protocol is not live. For protocol $\mathscr{P}$, let $\mathscr{L}_P$ be the conjunction of all the above property clauses. Our decision procedure is to check if $\mathscr{C}_P \wedge \mathscr{E}_P$ is satisfiable and $\mathscr{C}_P \wedge \mathscr{L}_P$ is unsatisfiable.

THEOREM 4. A protocol $\mathscr{P}$ is *live* if and only if $\mathscr{C}_P \wedge \mathscr{E}_P$ is satisfiable and $\mathscr{C}_P \wedge \mathscr{L}_P$ is *not* satisfiable.

Proof Sketch. Let τ satisfy $\mathscr{C}_P \wedge \mathscr{L}_P$. Then, by Theorem 1, we can construct a viable history vector that cannot be extended by a message emission (maximality) or by message reception (lossless transmission), and yet is incomplete. That is, such a history vector belongs to the universe of enactments of $\mathscr{P}$ but is neither complete nor can be completed. Thus, by Definition 15, $\mathscr{P}$ is not live.

Conversely, if a protocol $\mathscr{P}$ is *live*, we know there is a history vector in the universe of enactments of $\mathscr{P}$ that is complete. From Theorem 2, that vector satisfies $\mathscr{C}_P \wedge \mathscr{E}_P$. We also know that each history vector in the universe of enactments of $\mathscr{P}$ is either complete or can be finitely extended to a complete history vector. Thus if $\mathscr{P}$ is live, $\mathscr{C}_P \wedge \mathscr{L}_P$ is *not* satisfiable. □

6. DISCUSSION

We employ a satisfiability (SAT) solver for Precedence built using a propositional SAT solver—an established technology for logical reasoning. Our approach expresses and solves temporal constraints directly instead of building an explicit state machine representation. Table 1 shows the numbers of Precedence clauses for selected protocols.

Table 1: Counts of Precedence clauses for our properties.

Protocol	$\lvert\mathscr{C}_P\rvert$	$\lvert\mathscr{E}_P\rvert$	$\lvert\mathscr{S}_P\rvert$	$\lvert\mathscr{L}_P\rvert$	Total
Abrupt Cancel	14	2	4	9	29
Purchase	70	4	4	21	99
Purchase No Reject	56	4	0	19	79
Purchase No Ship	62	4	4	19	89
Purchase Unsafe	64	4	2	21	91
FIPA Request [6]	121	3	56	27	207

A technically correct protocol may have design flaws: (1) it may have deadwood messages, which can never be enacted; (2) parameters that may never be bound (e.g., a private parameter that is not ⌜out⌝ in any message); (3) some parameters that may never be used (e.g., a private parameter that is not adorned ⌜in⌝ in any message). Such checks are valuable because they indicate other problems.

We elide other easy checks that help validate a protocol but which are not critical to our verifier. A protocol that has a public ⌜in⌝ parameter cannot be enacted standalone: its intension is empty. A protocol that lacks a message involving any of its public ⌜out⌝ parameters is also not enactable.

Traditional notations for protocols such as AUML [11], UML 2.0 message-sequence charts (MSCs), and choreography description languages take a procedural stance for describing interactions. Thus they emphasize explicit constraints on how messages are ordered. Desai and Singh [4] identify several challenges to the enactability of a protocol: ordering problems termed *blindness* and occurrence problems termed *nonlocal choice* [7]. Traditional approaches formalize properties such as safety and liveness but those are understood purely procedurally and the underlying model does not sustain a declarative information-based model as BSPL does. In contrast, BSPL's parameter adornments force clarity in terms of causality and the flow of information. In this way, BSPL avoids both blindness and nonlocal choice: each of them yields an empty intension and is thus deemed nonenactable. A designer can correct an unsound protocol by inserting suitable messages.

Miller and McGinnis [9] propose RASA, a language for expressing protocols. RASA takes a procedural stance on capturing protocols and takes its semantics from propositional dynamic logic (PDL). RASA does not have a notion of parameter adornment as in BSPL and its semantics does not capture the ideas of maximizing concurrency and interaction orientation. Like BSPL, RASA supports protocols that an agent can inspect and reason about.

LoST [14] focuses on the architectural aspects of realizing a language such as BSPL. That work describes the functioning of a suitable middleware using conceptually relational data stores. It does not describe the formal semantics as introduced in this paper.

Several researchers have developed approaches for analyzing protocols [1, 5, 10, 15] that by and large consider higher-level aspects of interaction than BSPL. It would be interesting and useful to see how such approaches can be applied on top of BSPL. A potential advantage, and one that motivates BSPL, is that by guaranteeing the integrity of distributed enactments, BSPL can facilitate expressing high-level, declarative meanings of protocols, thereby facilitating analyses carried out in the above works. Thus the reasoning they perform on commitments, delegation, and other normative constructs could have even better effect than presently, in particular, obtaining a distributed rendition for free and thus the opportunity to apply in an asynchronous information-driven environment. Exploring such connections is an important theme for future research.

Acknowledgments

This work was partially supported by the OOI Cyberinfrastructure program, which is funded by NSF contract OCE-0418967 with the Consortium for Ocean Leadership via the Joint Oceanographic Institutions. Thanks to Matthew Arrott, Amit Chopra, and Claudiu Farcas for discussions. Thanks to Amit Chopra, Neil Yorke-Smith, and the anonymous reviewers for comments on previous versions.

7. REFERENCES

[1] A. Artikis, M. J. Sergot, and J. Pitt. An executable specification of a formal argumentation protocol. *Artificial Intelligence*, 171(10–15):776–804, 2007.

[2] J. L. Austin. *How to Do Things with Words*. Clarendon Press, Oxford, 1962.

[3] N. Desai and M. P. Singh. A modular action description language for protocol composition. In *Proc. 22nd Conference on Artificial Intelligence (AAAI)*, pp. 962–967, Vancouver, July 2007.

[4] N. Desai and M. P. Singh. On the enactability of business protocols. In *Proc. 23rd Conference on Artificial Intelligence (AAAI)*, pp. 1126–1131, Chicago, July 2008.

[5] M. El Menshawy, J. Bentahar, H. Qu, and R. Dssouli. On the verification of social commitments and time. In *Proc. AAMAS*, pp. 483–490, Taipei, May 2011.

[6] FIPA. FIPA interaction protocol specifications, 2003. FIPA: The Foundation for Intelligent Physical Agents, http://www.fipa.org/repository/ips.html.

[7] P. B. Ladkin and S. Leue. Interpreting message flow graphs. *Formal Aspects of Computing*, 7(5):473–509, Sept. 1995.

[8] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.

[9] T. Miller and J. McGinnis. Amongst first-class protocols. In *Proc. 8th International Workshop on Engineering Societies in the Agents World (ESAW 2007)*, *LNCS* 4995, pp. 208–223. Springer, 2008.

[10] T. J. Norman and C. Reed. A logic of delegation. *Artificial Intelligence*, 174(1):51–71, Jan. 2010.

[11] J. Odell, H. V. D. Parunak, and B. Bauer. Representing agent interaction protocols in UML. In *Proc. 1st International Workshop on Agent-Oriented Software Engineering (AOSE 2000)*, *LNCS* 1957, pp. 121–140. Springer, 2001.

[12] M. P. Singh. Distributed enactment of multiagent workflows: Temporal logic for service composition. In *Proc. AAMAS*, pp. 907–914, Melbourne, July 2003.

[13] M. P. Singh. Information-driven interaction-oriented programming: BSPL, the blindingly simple protocol language. In *Proc. AAMAS*, pp. 491–498, Taipei, May 2011.

[14] M. P. Singh. LoST: Local state transfer—An architectural style for the distributed enactment of business protocols. In *Proc. 7th IEEE International Conference on Web Services (ICWS)*, pp. 57–64, Washington, DC, 2011.

[15] P. Yolum. Design time analysis of multiagent protocols. *Data and Knowledge Engineering Journal*, 63:137–154, 2007.

Author Index

Abdallah, Sherief, 1381
Adams, Julie, 569, 593
Agmon, Noa, 341, 1251
Ågotnes, Thomas, 1099
Alan, Perotti, 1023
Alberola, Juan M., 1379
Alberti, Marco, 1425
Albrecht, Stefano, 349
Alcântara, João, 501
Aldewereld, Huib, 1371, 1421
Alechina, Natasha, 1057, 1099, 1309
Alers, Sjriek, 1475
Alferes, José, 1423
Alford, Ron, 981
Allahverdyan, Armen, 1391
Almajano, Pablo, 1483
Alvarado, Oscar, 1493
Amarante, Maicon, 1351
Amor, Mercedes, 1427
An, Bo, 13, 863, 1307
Anand, Sarabjot Singh, 1367
Andrighetto, Giulia, 1189
Anshelevich, Elliot, 1321
Antos, Dimitrios, 55
Aravamudhan, Ajay Srinivasan, 1227
Argente, Estefanía, 1419
Artstein, Ron, 63
Athakravi, Duangtida, 1369
Atkinson, Katie, 1171
Aumann, Yonatan, 1293
Ayala, Inmaculada, 1427
Aylett, Ruth, 1197, 1457
Azar, Yossi, 897
Azaria, Amos, 459
Aziz, Haris, 585, 763, 1311

Bachrach, Yoram, 535
Bai, Aijun, 1215
Bai, Quan, 1459
Baier, Jorge, 997
Baldwin, Craig, 13
Banerjee, Bikramjit, 1441
Baptista, Márcia, 1175
Barlow, Gregory J., 1271
Barrera, Francisco, 129
Barrett, Samuel, 129, 357
Baumeister, Dorothea, 577
Bazzan, Ana, 1351, 1389, 1395
Becerik-Gerber, Burcin, 21, 1455
Bekris, Kostas, 247
Bench-Capon, Trevor, 1171
Benenson, Itzhak, 1453
Bergenti, Federico, 1435
Berzan, Constantin, 189
Bessiere, Christian, 1263
Beygelzimer, Alina, 1317
Bianchi, Reinaldo, 1395
Bída, Michal, 1469, 1477
Biec, Santiago, 1481
Bill-Clark, Luis, 1213
Birattari, Mauro, 139
Bistaffa, Filippo, 1461
Bloembergen, Daan, 1393, 1475
Blokzijl-Zanker, Michiel, 1207
Boella, Guido, 1023
Bölöni, Ladislau, 1345
Bonjean, Noélie, 1065
Bonzon, Elise, 1413
Booth, Richard, 493
Bošanský, Branislav, 905, 1301, 1473
Botia, Juan, 307
Botti, Vicent, 1355, 1377, 1419, 1493
Bou Ammar, Haitham, 383
Boureanu, Ioana, 1141
Bourgne, Gauvain, 1223
Boutilier, Craig, 737
Bowring, Emma, 1193
Brafman, Ronen, 1265
Brambilla, Manuele, 139
Brandl, Florian, 763
Brandts, Jordi, 1189
Bratman, Jeshua, 407
Brill, Markus, 585
Brito, Ismel, 1263
Brom, Cyril, 1469, 1477
Bromuri, Stefano, 1487
Brown, Matthew, 863
Brunskill, Emma, 1385
Budrene, Elena, 1337
Bügler, Max, 1475
Burnett, Chris, 1359

Caillou, Philippe, 1353
Caire, Giovanni, 1435
Caminada, Martin, 493
Campano, Sabrina, 1191
Čáp, Michal, 1473
Carnevale, Peter, 55
Castelfranchi, Cristiano, 1241
Cataldi, Mario, 1185
Cavallo, Ruggiero, 677
Cavedon, Lawrence, 1081, 1187
Ceppi, Sofia, 1323
Cerquides, Jesus, 1275
Chaganty, Arun Tejasvi, 391
Chakraborty, Nilanjan, 1161

Chalkiadakis, Georgios, 417, 779, 787, 1165
Chang, Yu-Han, 45, 1295, 1505
Chatterjee, Sreerupa, 1333
Chella, Antonio, 1065
Chen Hui, Ong, 29
Chen, Inn-Tung, 1277
Chen, Jianing, 163
Chen, Jie, 105
Chen, Xiaoping, 1215
Chen, Yiling, 889
Chen, Yin, 1437
Cheng, Shih-Fen, 1227
Chhabra, Meenal, 1321
Chi, Luyan, 1295
Chien, Steve, 105
Chli, Maria, 1285
Claes, Daniel, 147, 1495
Cobo, Luis C., 483
Cohen, Robin, 1363
Colby, Mitchell, 425
Colombo Tosatto, Silvano, 1023
Conte, Rosaria, 1189
Corapi, Domenico, 1369
Corruble, Vincent, 1191
Cossentino, Massimo, 1065
Cramer, Henriette, 1197
Crandall, Jacob, 399
Cranefield, Stephen, 1491
Criado, Natalia, 1419
Croitoru, Madalina, 1249

Dam, Hoa, 1433
Damiano, Rossana, 1185
Dannenberg, Roger, 205
Das, Sanmay, 1291, 1321
Dasgupta, Prithviraj, 121
Dastani, Mehdi, 331, 1057, 1133, 1373
d'Avila Garcez, Artur, 1023
Dayama, Pankaj, 703
De Giacomo, Giuseppe, 1031
de Jonge, Dave, 1415
de la Hoz, Enrique, 1259
de Lucena, Carlos, 1225
de Melo, Celso, 55
de Sevin, Etienne, 1191
De Vos, Marina, 1369
de Weerd, Harmen, 1195
del Val, Elena, 1355, 1429
Delle Fave, Francesco Maria, 289, 1467
Demiris, Yiannis, 1207
Derbinsky, Nate, 1387
desJardins, Marie, 315
Devlin, Sam, 433
Di Caro, Gianni, 1205, 1503
Di Loreto, Ines, 1449
Dickerson, John, 711
Dignum, Frank, 1181
Dignum, Virginia, 1183, 1371, 1421
Dimas, Joana, 1175
Dimopoulos, Yannis, 1413
DiRenzo, Joseph, 13
Dobson, Andrew, 247
Dolan, John, 105, 1213
Dorigo, Marco, 97, 139
Doshi, Prashant, 1039, 1243, 1507
Dovgan, Erik, 1485
Driessen, Kurt, 383
Ducatelle, Frederick, 1205
Dunne, Paul, 939
Duong, Quang, 441
Durfee, Ed, 323, 1277

Eck, Adam, 1221
Elidrisi, Mohamed, 1289
Elkind, Edith, 627, 787
Endriss, Ulle, 635
Enz, Sibylle, 1197
Epstein, Daniel, 1389
Erdélyi, Gábor, 627
Ermon, Stefano, 965
Esteva, Marc, 1483

Fabregues, Angela, 1481
Falcone, Rino, 1241
Faliszewski, Piotr, 577
Faltings, Boi, 1273
Fang, Fei, 1299
Fang, Hui, 1365
Farinelli, Alessandro, 417, 1461
Faus, Jaume, 1499
Feige, Uriel, 897
Feldman, Michal, 771, 897
Fernandez, Alberto, 1429
Fink, Andreas, 1417
Fischer, Felix, 585
Fischer, Klaus, 1479
Foss, Bjarne, 1169
Fragiadakis, Daniel, 1327
Frazier, Spencer, 1505
Fredericks, Gary, 1255
French, Tim, 1091
Fridman, Natalie, 1343
Frieder, Asaf, 1281
Friedman, Michal, 1267
Fu, Tuanjie, 1331
Fuentes, Lidia, 1427

Gal, Ya'akov (Kobi), 451
Galstyan, Aram, 1391
Gambardella, Luca, 1205, 1503
Gams, Matjaž, 1485

Ganesan, Vijayalakshmi, 1403
Ganzfried, Sam, 871
Gao, Debin, 29
García-Fornes, Ana, 1377, 1379
Gatti, Nicola, 813, 1323, 1325
Gauci, Melvin, 163
Gaur, Prateek, 391
Gelfand, Michele, 451
Gemrot, Jakub, 1469
Genovese, Valerio, 1023
Genter, Katie, 1251
Gerding, Enrico, 661, 669, 1323
Gerrior, Matthew, 1321
Ghanem, Amer, 551
Ghorbani, Amineh, 1421
Ghose, Aditya, 1433
Ghosh, Siddhartha, 1471
Gil-Quijano, Javier, 1353
Gini, Maria, 1211, 1289
Giret, Adriana, 1493
Giusti, Alessandro, 1503
Gleizes, Marie-Pierre, 1065
Goel, Sharad, 1319
Goldman, Claudia, 459
Gomes, Ana Sofia, 1425
Gomes, Carla, 965
Gonçalves, Ricardo, 1423, 1425
Goodie, Adam, 1243
Goranko, Valentin, 1123
Gordon, Geoff, 1227
Gotta, Danilo, 1435
Governatori, Guido, 1375
Graepel, Thore, 535
Gratch, Jonathan, 55, 63
Grau, Ricardo, 1401
Griffiths, Nathan, 1367
Grimaldo, Francisco, 1357, 1499
Groß, Roderich, 163
Großekathöfer, Ulf, 1177
Grossi, Davide, 805
Grubshtein, Alon, 1267
Grześ, Marek, 1237
Guo, Mingyu, 745
Gutierrez, Patricia, 273, 1263
Gutman, Avital, 719

Hacker, Severin, 467
Hafizoğlu, Feyza, 1349
Haghpanah, Yasaman, 315
Haim, Galit, 451
Han , The Anh, 559
Hanna, Nader, 79, 1463
Harland, James, 1443
Harrenstein, Paul, 585, 1311
Hasegawa, Takato, 795
Hashimoto, Naoyuki, 795
Hayes, Timothy, 21, 1455
Hazon, Noam, 879
Hendrix, Philip, 1337
Hennes, Daniel, 147, 1475, 1495
Hermann, Thomas, 1177
Hernández, Carlos, 997
Hervouet, Fabien, 1449
Hickmott, Sarah, 1163
Hoang, Trong Nghia, 155, 1233
Hoey, Jesse, 1237
Horvitz, Eric, 467, 889, 1329
Hossain, S. G. M. , 121
Hrstka, Ondřej, 37
Hsu, Jane Yung-jen, 1439
Hsu, Wynne, 215
Hu, Cuiyun, 1437
Huang, Lixing, 63

Iliev, Petar, 1115
Inoue, Katsumi, 1223
Iocchi, Luca, 1203
Isbell Jr., Charles L., 483
Ishowo-Oloko, Fatimah, 1167
Itoh, Hidenori, 1159
Iwasaki, Atsushi, 753, 795, 1305, 1327

Jacobson, Michael J., 79, 1463
Jain, Manish, 905
Jain, Shaili, 677
Jakob, Michal, 37, 1501
Jamroga, Wojciech, 1123, 1405
Jazizadeh, Farrokh, 21, 1455
Jeanpierre, Laurent, 1209
Jennings, Nick, 223, 231, 289, 417, 543, 669, 779, 1165, 1167, 1467, 1471
Jiang, Albert Xin, 1299, 1501
Jiang, Jie, 1371
Jiang, Siwei, 299
Jiang, Yichuan, 1331
Jih, Wan-rong, 1439
Joe, YongJoon, 1305
John, Richard, 1297
Johnson, Matthew, 1501
Jones, Andrew, 1141
Jonker, Catholijn, 1183
Julian, Vicente, 1379, 1493
Junges, Robert, 1335

Kaelbling, Leslie, 973
Kaisers, Michael, 947, 1393
Kalia, Anup, 1489
Kaluža, Boštjan, 955, 1485
Kalyanakrishnan, Shivaram, 129
Kamar, Ece, 467, 1329
Kaminka, Gal, 113, 955, 1343

Kandori, Michihiro, 1305
Kang, Sin-Hwa, 63
Kang, Yi-Lin, 1465
Kankanhalli, Mohan, 155
Kannan, Balajee, 1213
Kantor, George, 1213
Karnik, Aditya, 703
Kasneci, Gjergji, 535
Kato, Shohei, 1159
Kavulya, Geoffrey, 21, 1455
Kearns, Michael, 441
Keidar, Matan, 113
Kelaiah, Iwan, 79, 1463
Kerr, Reid, 1363
Keysermann, Matthias, 1197
Kianercy, Ardeshir, 1391
Kiekintveld, Christopher, 863
Kim, Eunkyung, 1295
Kim, Yoonheui, 1279
Kimmel, Andrew, 247
Klein, Laura, 21, 1455
Klein, Mark, 1259
Klenk, Matthew, 989
Klügl, Franziska, 1335
Klusch, Matthias, 1479
Knobbout, Max, 331
Knorr, Matthias, 1425
Knox, W. Bradley, 475
Koenig, Sven, 997
Kok, Eric, 1411
Kollar, Thomas, 1217
Komenda, Antonín, 1239, 1473, 1501
Kopp, Stefan, 1177
Kosinski, Michal, 535
Koster, Andrew, 517
Kot, Alex C., 1361
Kota, Ramachandra, 1165
Kraemer, Landon, 1441
Krampf, Johannes, 1487
Kraus, Sarit, 451, 459, 1281, 1297
Kriegel, Michael, 1457
Kudenko, Daniel, 433
Kuter, Ugur, 981, 989, 1201
Kwak, Jun-young, 21, 1455

Lagoudakis, Michail, 171
Lai, Darong, 1331
Laird, John, 1387
Lang, Fabian, 1417
Lang, Jérôme, 577, 585, 1247
Langford, John, 1317
Lau, Hoong Chuin, 257, 1227
Lau, Qiangfeng Peter, 215
Lazaric, Alessandro, 1325
Lee, Mong Li, 215
Leite, João, 1425
Leonetti, Matteo, 1203
Lespérance, Yves, 1031
Lesser, Victor, 1279
Letchford, Joshua, 1303
Lev, Omer, 611
Lewis, Bennie, 1219
Lewis, Michael, 1161
Lewis, Richard, 407
Li, Justin, 1387
Li, Minyi, 525
Liang, Han, 1261
Lin, Raz, 1281
Linkov, Igor, 1337
Lisý, Viliam, 1301, 1473
Littman, Michael, 947
Liu, Fan, 1459
Liu, Siyuan, 1361
Logan, Brian, 1057, 1309
Lombardo, Vincenzo, 1185
Lomuscio, Alessio, 1141
Lopez Carmona, Miguel Angel, 1259
Lopez-Mobilia, Adrian, 129
Lopez-Sanchez, Maite, 1483
Lorini, Emiliano, 1133
Low, Kian Hsiang, 105, 155, 1233
Luck, Michael, 1225
Luke, Sean, 197
Luo, Jian, 1015
Luštrek, Mitja, 1485
Lyle, Jeremy, 1441

MacAlpine, Patrick, 129
Machado , André, 1389
Magee, Liam, 1163
Maghami, Mahsa, 687
Magnenat Thalmann, Nadia, 1365
Mahdian, Mohammad, 1319
Maheswaran, Rajiv, 21, 45, 1295, 1297, 1299, 1455, 1505
Maleki, Sasan, 1471
Mamidi, Sunil, 45
Manenti, Lorenza, 1341
Mao, Hua, 1015
Mao, Xinjun, 1437
Marecki, Janusz, 821, 1235
Markakis, Vangelis, 779
Marsa-Maestre, Ivan, 1259
Marsella, Stacy, 1193
Martinho, Carlos, 1175
Mathews, Nithin, 97
Matignon, Laëtitia, 1209
Mattei, Nicholas, 1407
Maudet, Nicolas, 1223, 1313
Maule, Ben, 13
Mauri, Alessia, 1339

McClean, Sally, 417
Meeussen, Wim, 147, 1495
Mehrotra, Siddharth, 1161
Meir, Reshef, 771
Meisels, Amnon, 1267
Meneguzzi, Felipe, 1161
Meseguer, Pedro, 273, 1263
Męski, Artur, 1447
Meyer, Garrett, 13
Meyer, John-Jules, 1411
Miao, Chunyan, 1361, 1465
Michalak, Tomasz, 239
Migeon, Frédéric, 1065
MikoŁaj, Mikolaj, 493
Miles, Simon, 1225
Miller, Sam, 281
Minai, Ali, 551
Moler, Zbyněk, 1501
Molineaux, Matthew, 989
Moraitis, Pavlos, 1413
Morel, Benoit, 1337
Morency, Louis-Phillippe, 63
Mouaddib, Abdel-Illah, 1209
Mouri, Takayuki, 753
Muise, Christian, 1031
Mutoh, Atsuko, 1159

Nagi, Jawad, 1503
Nahum, Yinon, 1291
Nakisaee, Ali, 1497
Nanjanath, Maitreyi, 1211
Narahari, Yadati, 703
Narayanan, Ajit, 1459
Natarajan, Prabhu, 155
Nau, Dana, 981, 1201
Nelson, Carl, 121
Newnan, Alex, 1505
Newstead, Anne, 79, 1463
Ng, Wing Lon, 653
Nguyen, Duc Thien, 257
Nguyen, Nhan-Tam, 1287
Nguyen, Trung Thanh, 1287
Ning, Yu, 1295
Nisan, Noam, 719
Nissim, Raz, 1265
Noriega, Pablo, 1419, 1421
Norman, Timothy, 1409
Novák, Peter, 1239, 1473
Nowé, Ann, 1401
Nunes, Ernesto, 1211
Nunes, Ingrid, 1225

Obara, Ichiro, 1305
Ochs, Magalie, 87
Ogawa, Yuki, 1347
Oh, Jean, 1161
Ohta, Naoki, 795
Okada, Isamu, 1347
Okamoto, Steven, 879
Oliehoek, Frans, 973, 1229
Onaindia, Eva, 509
Ong, Yew-Soon, 299, 1465
Ordóñez, Fernando, 847, 863
Oren, Nir, 1359
Ossowski, Sascha, 1173
Othman, Abraham, 645
Ottens, Brammert, 1273

Padget, Julian, 1369
Padgham, Lin, 1049, 1081, 1163, 1187
Pajares Ferrando, Sergio, 509
Palit, Imon, 653
Pan, Yinghui, 1015
Panozzo, Fabio, 813
Paolucci, Mario, 1357
Paraschos, Alexandros, 171
Pardo, Pere, 1231
Parkes, David, 889
Parr, Gerard, 417
Pěchouček, Michal, 37, 905, 1239, 1301, 1473, 1501
Pedersen, Sindre, 1169
Pelachaud, Catherine, 87, 1179
Peled, Hilla, 265
Peña, Jorge, 1175
Penczek, Wojciech, 1447
Pennock, David, 1317, 1319
Penya-Alba, Toni, 1275
Pereira , Luís Moniz, 559
Perez, Victor, 1493
Pham, Manh Tung, 1257
Phelps, Steve, 653
Pinciroli, Carlo, 139
Pini, Maria Silvia, 1313, 1407
Pita, James, 1297
Piunti, Michele, 1241
Pizzo, Antonio, 1185
Popelová, Markéta, 1477
Porte, John, 79, 1463
Prada, Rui, 1175
Prakken, Henry, 1411
Prepin, Ken, 1179
Price, Michael J., 163
Procaccia, Ariel, 711
Proper, Scott, 1397
Pryymak, Oleksandr, 543
Pyrga, Evangelia, 1311

Qu, Xia, 1243

Rabinovich, Zinovi, 459
Raboin, Eric, 1201
Rahwan, Iyad, 493, 1167

Rahwan, Talal, 239
Raimundo, Guilherme, 1175
Rajagopal, Karthik, 1299
Ramamoorthy, Subramanian, 349, 1203
Ramchurn, Sarvapali, 223, 281
Ranathunga, Surangika, 1491
Ranjbar-Sahraei, Bijan, 1497
Rao, Karun, 375
Ravindran, Balaraman, 391
Read, Stephen, 55
Rebollo, Miguel, 1355, 1429
Reeves, Daniel, 1319
Reijngoud, Annemieke, 635
Ribeiro, Luís Landeiro, 1175
Richards, Deborah, 79, 1463
Riedl, Mark, 71
Robu, Valentin, 669, 1165
Rodríguez, Abdel, 1401
Rodriguez, Inmaculada , 1483
Rodriguez, Rosa Maria, 1493
Rodriguez-Aguilar, Juan Antonio, 1275
Rogers, Alex, 281, 289, 417, 543, 661, 1165, 1461, 1467, 1471
Roos, Magnus, 1287
Rose, Harry, 661
Rosenschein, Jeffrey, 611, 1315
Rossi, Francesca, 1313, 1407
Rothe, Jörg, 577, 1287
Rovatsos, Michael, 307
Rudolph, Sebastian, 1249
Ruff, Alexander, 1333
Rui, Zaojie, 1331
Ruiz, N., 1493
Russo, Alessandra, 1369

Sá, Samy, 501
Sabater-Mir, Jordi, 517, 1189
Sabouret, Nicolas, 1191
Sadat, Seyed Abbas, 1199
Sadrzadeh, Mehrnoosh, 1231
Saffidine, Abdallah, 1247
Samadi, Mehdi, 1217
Sandholm, Tuomas, 645, 711, 729, 871
Santos, Francisco C., 559
Santos, Pedro A., 1175
Sardina, Sebastian, 1049, 1081
Sarne, David, 1291, 1293, 1445
Satoh, Ken, 1369
Sattar, Abdul, 1375
Scerri, Adrian, 1213
Scerri, Paul, 1213, 1261, 1269
Scheidler, Alexander, 97
Scheutz, Matthias, 189
Schjølberg, Ingrid, 1169
Schorlemmer, Marco, 517
Schumacher, Michael, 1487
Schumann, René, 1487
Seedig, Hans Georg, 585
Segal, Richard, 821
Seidita, Valeria, 1065
Selman, Bart, 965
Sen, Sandip, 1333, 1349
Şensoy, Murat, 1365
Seow, Kiam Tian, 1257
Serrano, Emilio, 307, 1377
Service, Travis, 569, 593
Shankar, Kumar Shaurya, 1213
Shapiro, Steven, 1081
Shehory, Onn, 1291
Shieh, Eric, 13
Shivashankar, Vikas, 981
Sidner, Candy, 63
Sierra, Carles, 1415, 1481
Singh, Munindar, 1073, 1149, 1489
Singh, Satinder, 407, 441, 1277
Slavkovik, Marija, 1403, 1405
Sleight, Jason, 323
Slota, Martin, 1425
Slusallek, Philipp, 1479
Smith, Stephen, 1271
Sofy, Nadav, 1445
Soh, Leen-Kiat, 1221
Song, Zhao, 1199
Sonu, Ekhlas, 1039, 1507
Sørensen, Troels, 829
Sorg, Jonathan, 407
Sousa, Sergio, 1403
Spaan, Matthijs, 1229
Spanoudakis, Nikolaos, 171
Sridharan, Mohan, 181
Stein, Sebastian, 231, 669
Ştiurcă, Nicolae, 129
Stone, Peter, 129, 341, 357, 475, 1251
Stranders, Ruben, 289
Stranieri, Alessandro, 97
Subagdja, Budhitama, 1007, 1465
Such, José, 1377
Suetsugu, Katsuya, 1159
Sukthankar, Gita, 687, 1219
Sullivan, Keith, 197
Sutanto, Danny, 1253, 1431
Sycara, Katia, 879, 1161, 1269, 1409
Szczepański, Piotr, 239
Szreter, Maciej, 1447

Tambe, Milind, 13, 21, 847, 855, 863, 905, 955, 1193, 1297, 1299, 1307, 1455, 1501
Tan, Ah-Hwee, 1007, 1465
Tan, Yao-Hua, 1371
Tan, Yuan-Sin, 1007

Tang, Pingzhong, 729
Tay, Junyun, 205
Taylor, Charlotte, 79, 1463
Taylor, Matthew, 383, 1383
Taylor, Meredith, 79, 1463
Teacy, Luke, 417
Telang, Pankaj, 1073, 1489
Tennenholtz, Moshe, 771, 897
Teow, Loo-Nin, 1007
Tesauro, Gerry, 821
Tettamanzi, Andrea, 1339
Thakur, Subhasis, 1375
Thangarajah, John, 1049, 1081, 1187, 1443
Theng, Yin-Leng, 1361
Thom, James, 1163
Thomaz, Andrea, 483
Thompson, David, 105
Tittle, James, 1161
Tjønnås, Johannes, 1169
Todo, Taiki, 753
Tomaszewski, Christopher, 1213
Tomek, Jakub, 1477
Toniolo, Alice, 1409
Torrey, Lisa, 1383
Tran-Thanh, Long, 289
Trescak, Tomas, 1483
Trodd, Luke, 1443
Troquard, Nicolas, 1245
Trovò, Francesco, 1325
Troyan, Peter, 1327
Tsai, Jason, 1193
Tsao, Tiffany Yi-Ting, 1439
Tsimhoni, Omer, 459
Tumer, Kagan, 425, 1397
Turrini, Paolo, 805
Tuyls, Karl, 147, 383, 1393, 1475, 1495

Ueda, Suguru, 795, 1327
Ufimtsev, Vladimir, 121
Uras, Tansel, 997
Urieli, Daniel, 129

Valada, Abhinav, 1213
van der Hoek, Wiebe, 1115
van der Schaar , Mihaela, 1283
van der Torre, Leendert, 1023, 1373, 1403
van der Zwaan, Janneke, 1183
van Ditmarsch, Hans, 1091, 1247
Van-Gael, Jurgen, 535
van Oijen, Joost, 1181
Vaněk, Ondřej, 37, 905
Varakantham, Pradeep, 21, 29, 1227, 1235, 1269, 1455
Vargas, Patricia, 1197
Vasirani, Matteo, 1173, 1429
Vaughan, Richard T., 1199
Vedanarayanan, Srinivasa, 551
Velagapudi, Prasanna, 1213, 1269
Velázquez-Quesada, Fernando R., 1091
Veloso, Manuela, 205, 1217
Venable, Kristen Brent, 1313, 1407
Venanzi, Matteo, 1241
Verbrugge, Rineke, 1195
Verheij, Bart, 1195
Verwer, Sicco, 1399
Vidal, José, 1255
Villatoro, Daniel, 1189
Vinyals, Meritxell, 1275, 1461
Vizzari, Giuseppe, 1341
Vladimirsky, Alexander, 965
Vo, Bao, 525
Voice, Thomas, 223
Vorobeychik, Yevgeniy, 1303, 1307
Vrancx, Peter, 1401
Vreeswijk, Gerard, 1411
Vu, Victor, 129
Vytelingum, Perukrishnen, 1167

Waizman, Gennady, 1453
Wallace, Iain, 1457
Walsh, Toby, 603
Wang, Wenwen, 1007
Warwas, Stefan, 1479
Waugh, Kevin, 871
Weiss, Gerhard, 383, 1497
Wellman, Michael, 441, 931
Whiteson, Shimon, 375
Wiedenbeck, Bryce, 931
Williamson, Simon, 29, 231
Winsper, Michael, 1285
Witwicki, Stefan, 973, 1277
Wöhler, Nils-Christian, 1177
Wong, Wilson, 1187
Wood, Wendy, 21, 1455
Wooldridge, Michael, 939, 1115
Woźna-Szcześniak, Bozena, 1447
Wu, Feng, 1215
Wunder, Michael, 947
Wyner, Adam, 1171

Xia, Guangyu, 205
Xia, Lirong, 603
Xie, Xiao-Feng, 1271
Xu, Jie, 1283
Xu, Yang, 1261

Yamamoto, Hitoshi, 1347
Yang, Rong, 13, 847, 1297, 1299
Yaros, John Robert, 947
Ye, Dayong, 1253, 1431
Yeoh, William, 257, 1227, 1269
Yin, Zhengyu, 855, 905, 1501
Yokoo, Makoto, 753, 795, 1305, 1327

Yorke-Smith, Neil, 1373
Yu, Hong, 71

Zame, William, 1283
Zappala, Julian, 1309
Zbrzezny, Andrzej, 1447
Zeng, Yifeng, 1015
Zhang, Haoqi, 889
Zhang, Jie, 299, 1365
Zhang, Minjie, 1253, 1431
Zhang, Shiqi, 181
Zhang, Yingqian, 1399
Zhou, Huiping, 1437
Zhu, Linglong, 1261
Zick, Yair, 787
Zilka, Avishay, 1343
Zivan, Roie, 265, 1267
Zoll, Carsten, 1197
Zuckerman, Michael, 1315